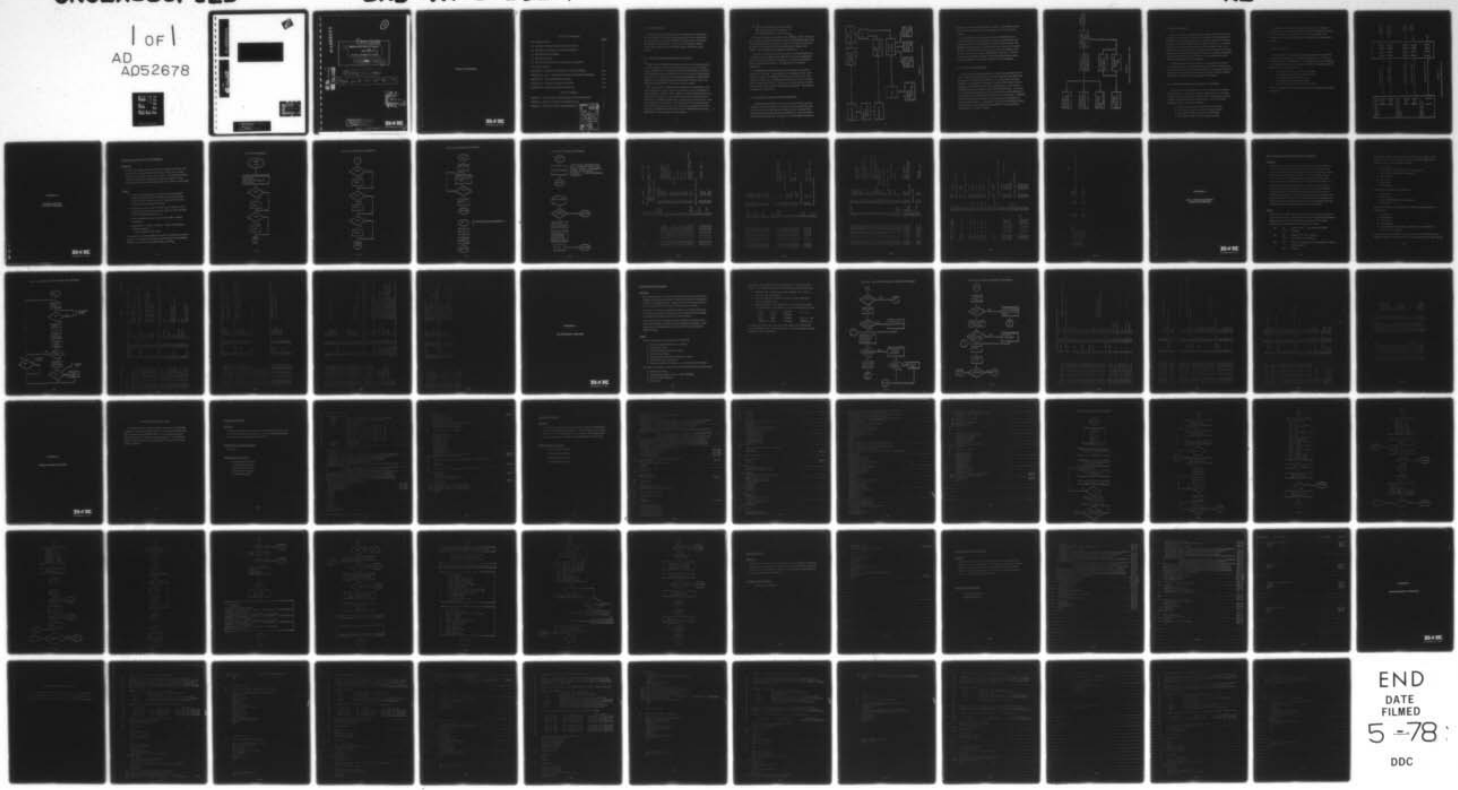
AD-A052 678 B-K DYNAMICS INC ROCKVILLE MD F/G 9/2
QUARTERLY INTERIM TECHNICAL REPORT (2ND), CONTRACT DAAH01-75-C---ETC(U)
APR 75 DAAH01-75-C-0194
UNCLASSIFIED BKD-TR-3-192 NL

1 OF 1
AD
AD52678



END
DATE FILMED
5-78
DDC

DISTRIBUTION STATEMENT A

Approved for public release;
Distribution Unlimited

(1)

(14) BKD – TR-3-192

(6) ~~2nd~~ Quarterly Interim Technical Report (2nd),
Contract DAAHØ1-75-C-Ø194

(11) 15 Apr 75 (12) 89 p.

(15) DAAHØ1-75-C-Ø194

(9) Rept. for 15 Jan – 15 Apr 75.

387 575

BOK DYNAMICS, INC.

TABLE OF CONTENTS

BOK
DYNAMICS, INC.

TABLE OF CONTENTS

LIST OF FIGURES

1.0 INTRODUCTION

B-K Dynamics' activities during the second quarter (15 January 1975 to 15 April 1975) have been focused on two areas: preparation of hardware for the interim STINGER simulation and conversion of the STINGER simulation to the ASC's hybrid computer system. This report summarizes the work performed during the above period.

2.0 INTERIM STINGER SIMULATION ACTIVITIES

BKD's interim STINGER simulation activities have been associated with readying the hardware for data transmission between the major system components (i.e. SDS/9300 to AD-4 and SDS/9300 to GE/3010). AD-4 linkage hardware, including the AD-4 converter and hybrid interface unit, was exercised and evaluated for performance. Sample/hold amplifiers were calibrated and data transmitted from the AD-4 to the IRSS using the SDS/9300. These tests were successful.

During the quarter, IRSS interface operation was intermittent. Extensive testing was done to discover the source of errors generated in data exchanges. Grounds, power supplies and other potential sources of noise were investigated. The problem was finally resolved by the replacement of several marginal gates, relocation of driver/receiver cards to the GE/3010's CPU chassis, and modification of the ground system. In the course of the above activities, software was generated for use in checking the SDS/9300 to GE/3010 link and the SDS/9300 to AD-4 link. Specifically, three programs were developed;

- A general purpose AD-4 test routine,
- An AD-4 discrete test routine, and
- A GE/3010 interface test routine.

The general purpose AD-4 test routine contains an ADC read loop and a register read/write loop along with three additional buffer areas for storing programs for the AD-4. Code for the AD-4 is entered in the buffers and executed from the SDS/9300 under sense switch control. The program is documented in Appendix A.

The AD-4 discrete test routine provides a convienent method for verifying that the 16 input and 16 output lines between the SDS/9300 and the AD-4 are operating. In addition it verifies the operation of DGS's and DGC's on the AD-4. The program is described in Appendix B.

The SDS/9300 to GE/3010 interface test program verifies the operation of that link. The program has two options for data transmitted. A count from 000000_8 to 177777_8 in increments of 1 bit is normally transmitted and optionally a pattern of all ones alternating with all zeroes may be sent. The data is transmitted from the SDS/9300 to the GE/3010 and then read back and compared. The program is described in Appendix C.

3.0 STINGER CONVERSION TO ASC EQUIPMENT

Converting the STINGER real-time simulation from the interim system to the ASC equipment has required replacing IBM-7094 software functions with equivalent CDC-6600 functions, and IBM-DOS interface operations with equivalent DADIOS ADC, DAC and discrete word handling capabilities. In Figure 1 the relationship of hardware

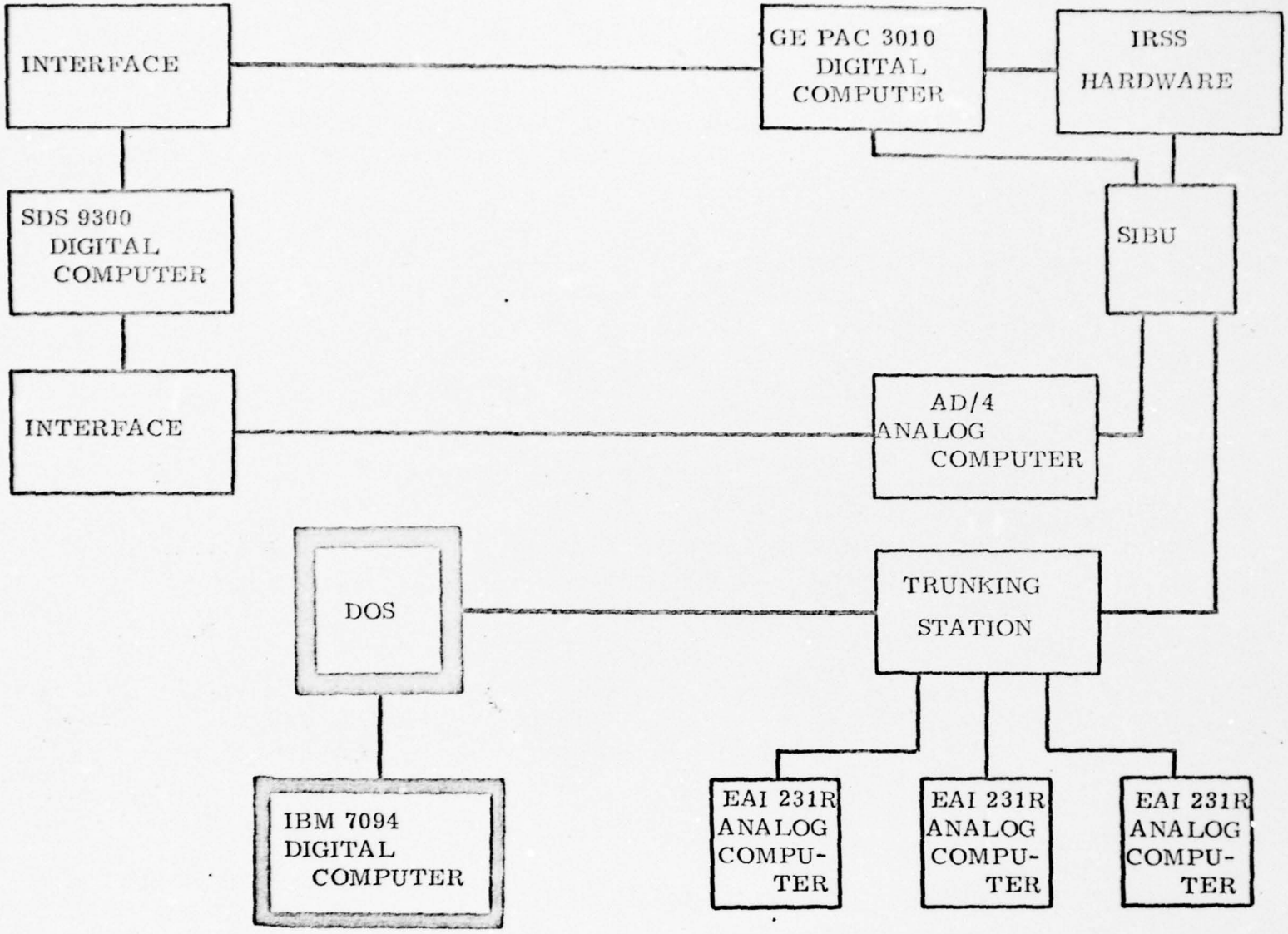


FIGURE 1

THE INTERIM STINGER REAL TIME SIMULATION

elements in the interim simulation are shown. The elements which are affected in this phase of the conversion are outlined in heavy borders.

To date STINGER software conversion and DADIOS checkout routines have both been completed. In addition a preliminary real-time I/O design has been developed based on results obtained from DADIOS checkout studies. The final real-time checkout will require extensive hardware/software test and the successful integration of the new software with the existing STINGER simulation. The flow chart in Figure 2 shows the interrelationship between these tasks. The tasks outlined in solid lines represent those completed.

3.1 SOFTWARE CONVERSION

The interim STINGER simulation has approximately 2500 lines of code written for the IBM/7094. Of this code approximately 2000 lines of code are in FORTRAN and 500 lines are in MAP (7094 assembly language). Converting this software to the CDC-6600 required minor changes to the FORTRAN code and a completely new code written in FORTRAN hybrid to replace the IBM-7094 assembly code. Conversion of the assembly language portion of the code has been accomplished by 1) generating a flow chart from the MAP code, 2) rewriting in FORTRAN hybrid the equilivent functions and 3) incorporating the appropriate real-time input/output. In Appendix D the equilivent FORTRAN hybrid code is given. The statements labeled SOFT-T are modifications pertaining to the software testing.

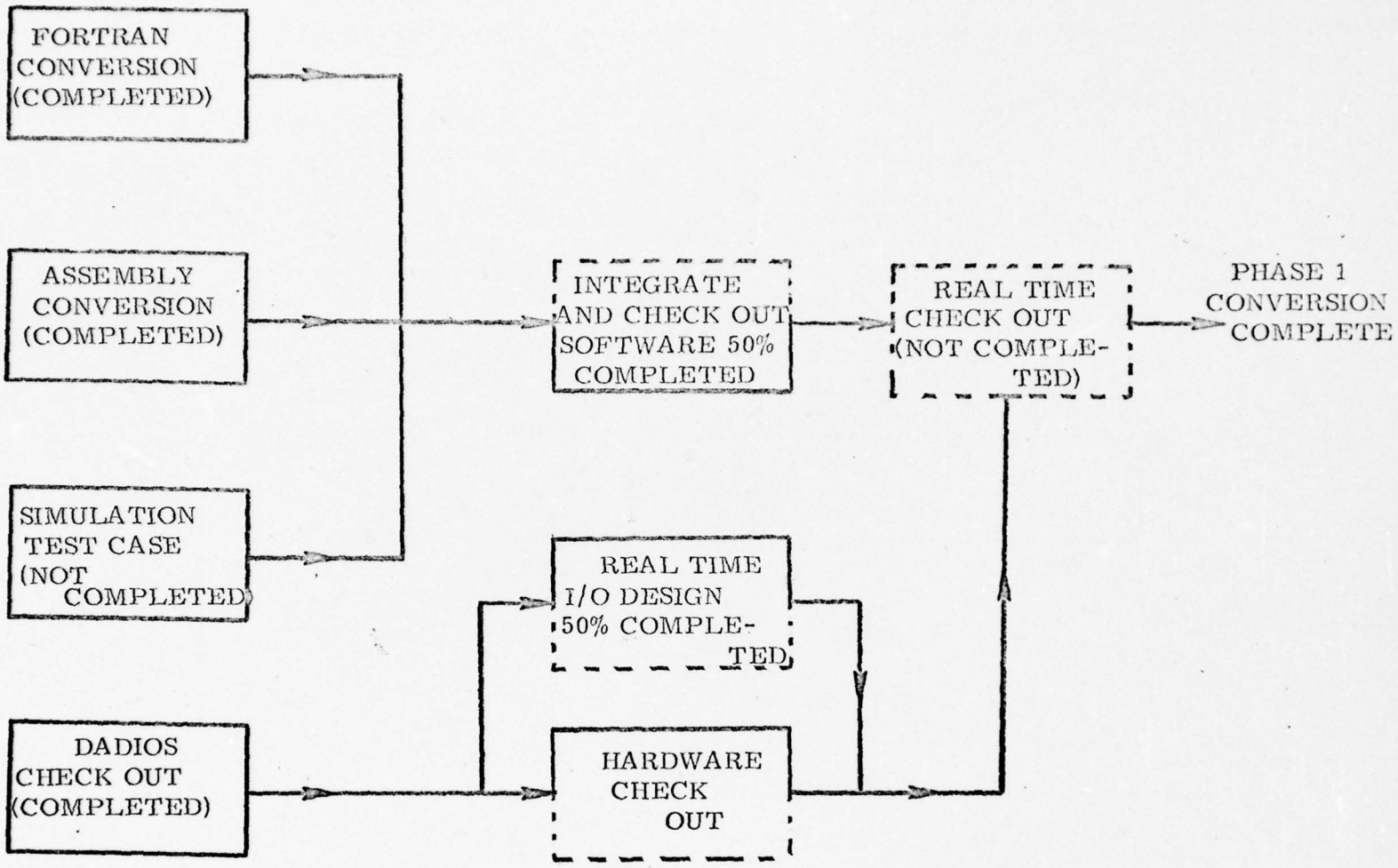


FIGURE 2

PHASE 1 STINGER CONVERSION TASK

3.2 SOFTWARE TEST

In order to reduce real-time software testing all new software will be tested in three steps: First, the new software will be simulated in non-real-time using special software to simulate real-time functions. Secondly, the software will be tested in pseudo real-time, a real-time test which uses only the most critical real-time loop. Then, if the previous tests are successful, the software will be tested in real-time. Evaluation of software test results will consist primarily of comparing interim STINGER data acquired from current simulation runs with results obtained from the new software.

In Appendix D the software necessary for simulating real-time events is presented. The special tasks which simulate hardware I/O are identified in card columns 73 through 80 by the designator SOFT-T. Other tasks which correct inconsistencys between the FTH. compiler and FTHH. are denoted by the designator SOFT-MOD.

3.3 SYSTEM SOFTWARE AND INTERFACE CHECKOUT

Prior to real-time simulation the system software and interface must be verified. This type of testing is important for 1) verifying status of the real-time system software and 2) checking the accuracy of ADCs, DACs and discrete communication. The test routines completed thus far include the following:

- Verification of discretes from AD-4 to CDC-6600.
- Verification of discretes from CDC-6600 to AD-4.
- Verification of ADCs from AD-4 to CDC-6600.
- Verification of DACs from CDC-6600 to AD-4.

In addition to establishing equipment status prior to real-time simulation, these test procedures are also used for isolating hardware or software failures in the interface system. In Appendix E the checkout programs are presented with an explanation of usage given in the code.

3.4 REAL-TIME I/O

The real-time I/O operations previously handled by the IBM-DOS are now implemented on the CDC-6600/DADIOS system. These new I/O task have been implemented in the FORTRAN hybrid code (Appendix D). These modifications will provide the STINGER simulation with the following hybrid linkage hardware:

- 16 Logic trunks from AD-4 to CDC-6600.
- 16 Logic trunks from CDC-6600 to AD-4.
- 16 DACs from CDC-6600 to AD-4.
- 16 ADCs from AD-4 to CDC-6600.
- 3 Interrupt lines from AD-4 to CDC-6600.

The AD-4 trunkline and ASFISS trunking station assignments are shown in Figure 3.

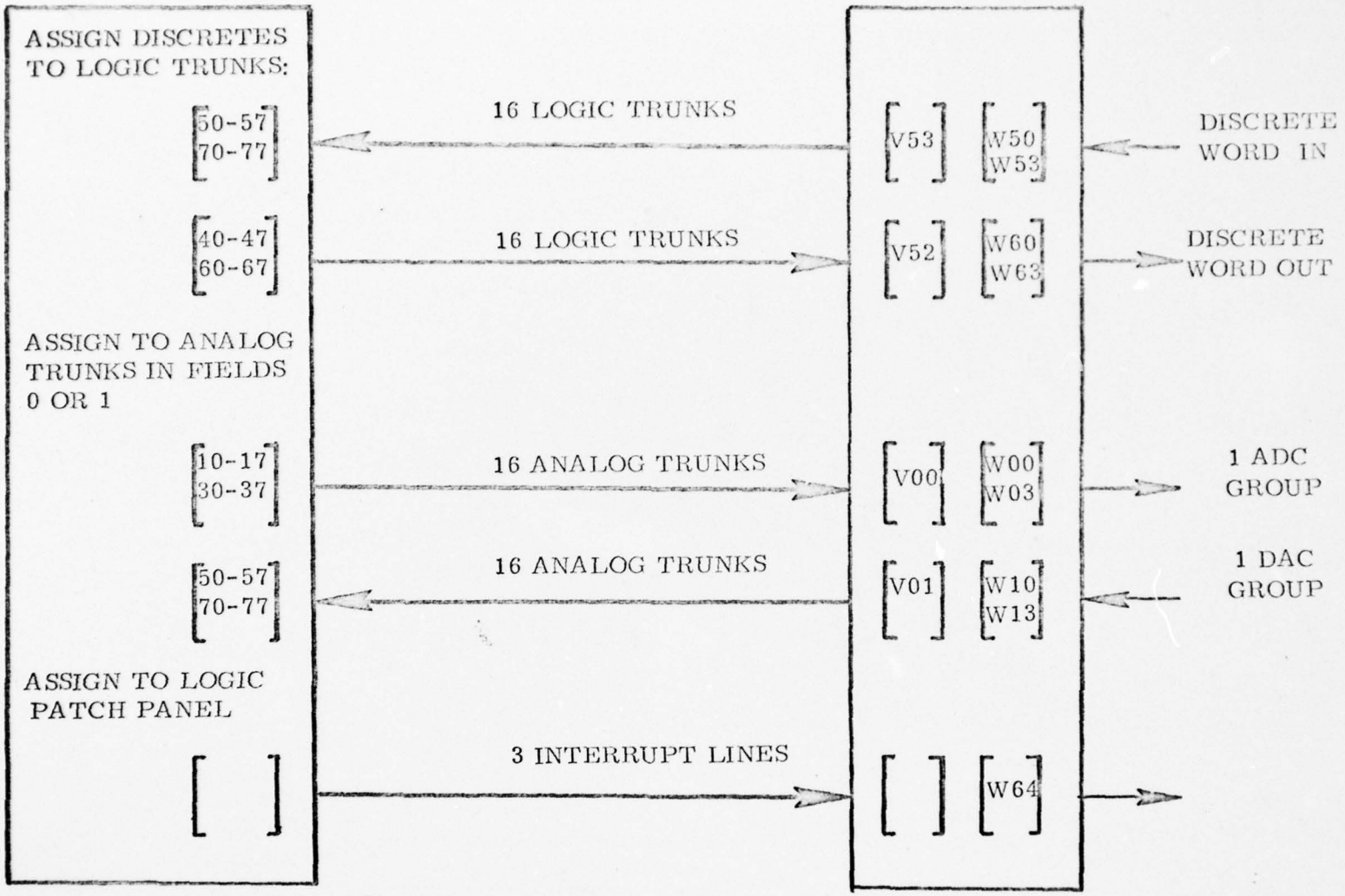


FIGURE 3

DADIOS PATCHING REQUIREMENTS

APPENDIX A

GENERAL PURPOSE AD/4 TEST PROGRAM

BOK
DYNAMICS, INC.

GENERAL PURPOSE AD-4 TEST PROGRAM

PURPOSE

The General Purpose AD-4 Test Program permits the user to execute up to five separate AD-4 routines (e.g. ADC read, register read/write etc.) under sense switch control. The program contains a register read/write routine and an ADC read routine. Three other program areas are available for the user to enter his own code.

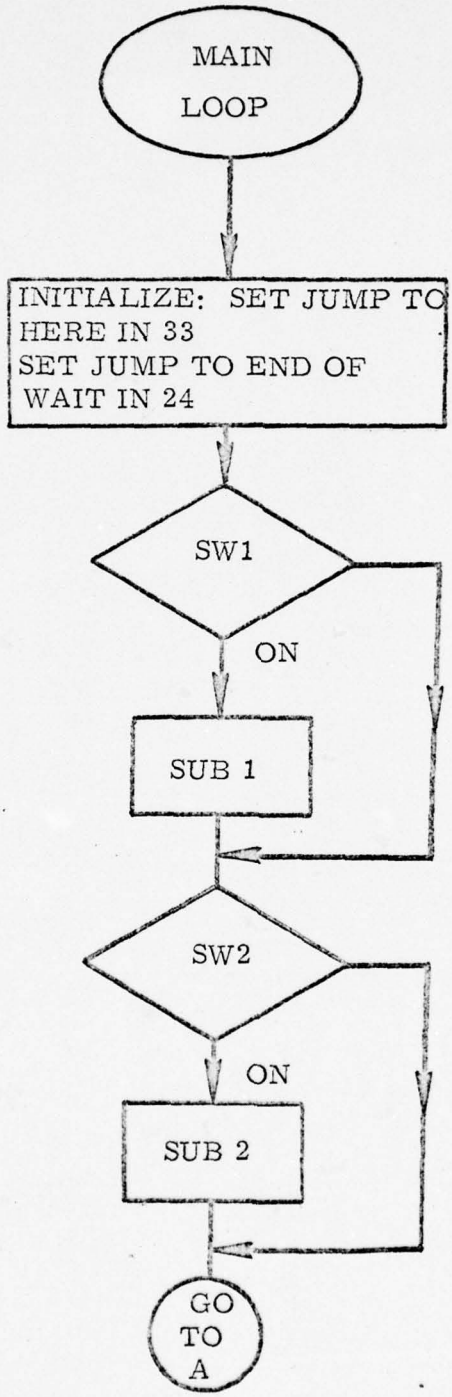
USAGE

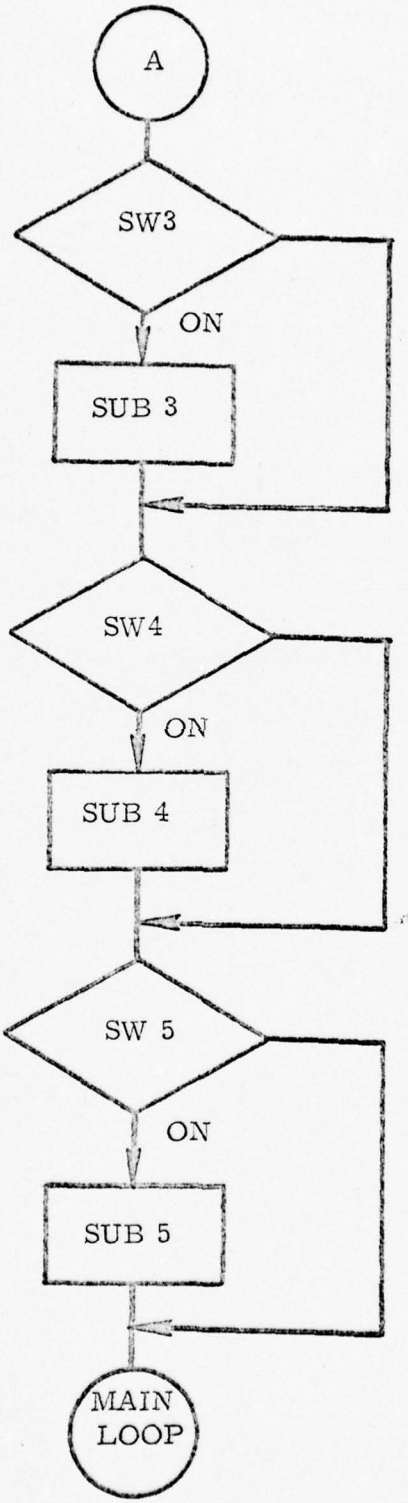
- Five AD-4 command lists are defined at locations 02100, 02200, 02400 and 02500. The first word of each list consists of the word count of that list. Subsequent words are the actual AD-4 command list words.
- If switch n is on, then list n will be sent. However, if the word count is LE 0 for a given list, that list is not sent even if the corresponding switch is on.
- New command lists may be added to the deck or loaded separately.
- OPERATION. Load the program. To run it perform the following sequence:

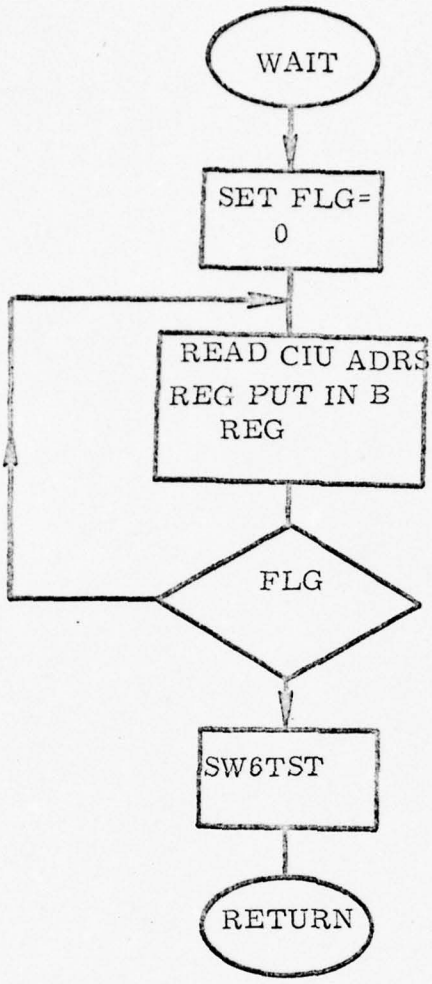
  IDLE, RESET, RUN, INT33

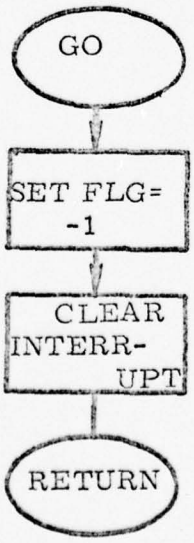
If one pass is desired set SS6 then set SSn for execution of the AD-4 program. The program will loop at location 01127 when complete. Results will be stored at locations 60000_8 to 60100_8.

AD/4 TEST PROGRAM

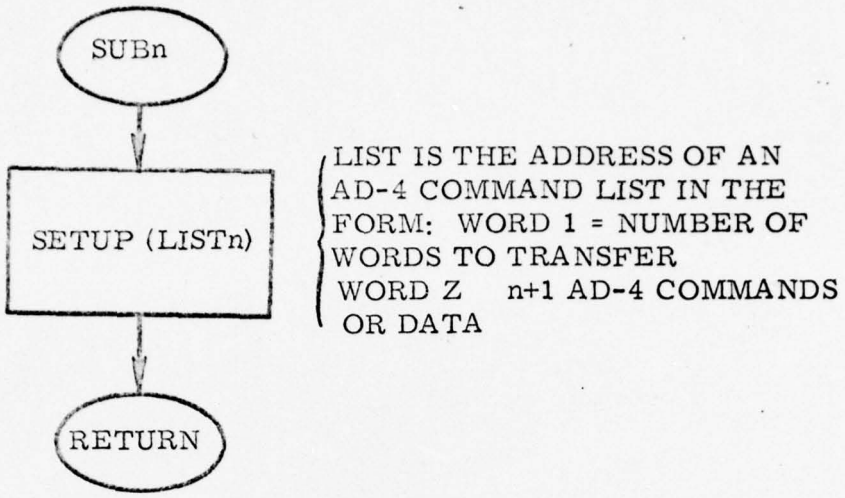


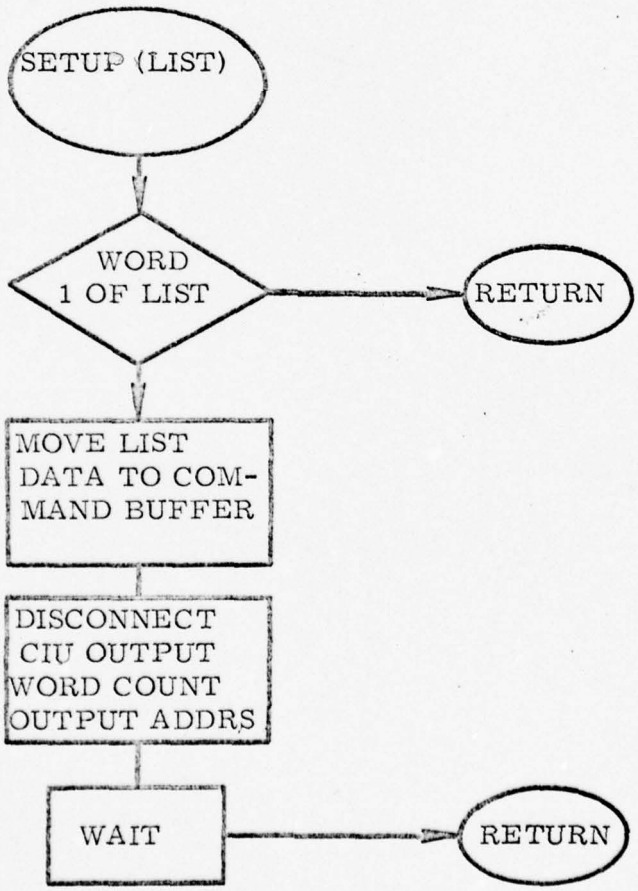

A
SW3
ON
SUB 3
SW4
ON
SUB 4
SW 5
ON
SUB 5
MAIN LOOP





{ENTERED FROM INTERRUPT 24


SUBn
SETUP (LISTn)
RETURN
LIST IS THE ADDRESS OF AN AD-4 COMMAND LIST IN THE FORM: WORD 1 = NUMBER OF WORDS TO TRANSFER
WORD Z n+1 AD-4 COMMANDS OR DATA


SETUP (LIST)
WORD 1 OF LIST
RETURN
MOVE LIST DATA TO COMMAND BUFFER
DISCONNECT CIU OUTPUT WORD COUNT OUTPUT ADDRS
WAIT
RETURN

```
01000                                   1          ABRG   01000          START AT 01000
                                        2 *
                                        3 *      AD/4 TEST
                                        4 *           WILL TRANSMIT FIVE DIFFERENT AD/4
                                        5 *           COMMAND LISTS TO THE AD/4
                                        6 *           COMMAND
                                        7 *           UNDER SWITCH CONTROL
                                        8 *
                                        9 EOM1     OPD   020276600
                                       10 EOM2     OPD   020276700
01000  0 0 16 01021                    11 STRT     LDA    TRP24           SET UP INTERRUPT
01001  0 0 76 00024                    12          STA    024               LOCATION 24
01002  0 0 16 01020                    13          LDA    TRP33           SET UP CONSOLE
01003  0 0 76 00033                    14          STA    033               INT 033
01004  00220002                        15          EIR                    ENABLE INT SYS
01005  0 224 0040                      16          SWT    040             CHK SW1
01006  0 0 03 01022                    17          BRM    SUB1
01007  0 224 0020                      18          SWT    020         CHK SW2
01010  0 0 03 01026                    19          BRM    SUB2
01011  0 224 0010                      20          SWT    010             CHK SW3
01012  0 0 03 01032                    21          BRM    SUB3
01013  0 224 0004                      22          SWT    004             CHK SW4
01014  0 0 03 01036                    23          BRM    SUB4
01015  0 224 0002                      24          SWT    002             CHK SW5
01016  0 0 03 01042                    25          BRM    SUB5
01017  0 0 10 00000                    26          NOP                    IN CASE A PATCH IS NEEDED
01020  0 0 01 01000                    27 TRP33    BRU    STRT            LOOP ALSO LOC 033
01021  0 0 03 01062                    28 TRP24    BRM    GO              LOC 024
                                       29 *
                                       30 *      SUBROUTINES FOR EACH LIST
                                       31 *
01022  0 0 00 00000                    32 SUB1     PZE    0
01023  0 0 03 01066                    33          BRM   ->SETUP          TRANSMIT LIST 1
01024  0 0 00 02100                    34          PZE    BUF1
01025  0 0 41 01022                    35          BRR    SUB1
01026  0 0 00 00000                    36 SUB2     PZE    0
01027  0 0 03 01066                    37          BRM    SETUP
01030  0 0 00 02200                    38          PZE    BUF2
```

```
01031  0 0 41 01026          39          BRR     SUB2
01032  0 0 00 00000          40 SUB3     PZE     0
01033  0 0 03 01066          41          BRM     SETUP
01034  0 0 00 02300          42          PZE     BUF3
01035  0 0 41 01032          43          BRR     SUB3
01036  0 0 00 00000          44 SUB4     PZE     0
01037  0 0 03 01066          45          BRM     SETUP
01040  0 0 00 02400          46          PZE     BUF4
01041  0 0 41 01036          47          BRR     SUB4
01042  0 0 00 00000          48 SUB5     PZE     0
01043  0 0 03 01066          49          BRM     SETUP
01044  0 0 00 02500          50          PZE     BUF5
01045  0 0 41 01042          51          BRR     SUB5

                             52 *
                             53 *        WAIT LOOP
                             54 *
01046  0 0 00 00000          55 FLG      PZE     0               FLAG TO COMMUNICATE
01047  0 0 00 00000          56 WAIT     PZE     0                 WITH INTRPT RTNE
01050  0 0 77 01046          57          STZ     FLG             SET FLG>0
01051  0 0 10 00000          58          NOP
01052  0 2 02 76702          59          EOM2    2       READ CIU ADRS CNTR
01053  0 0 33 01117          60          PIN     PNTR
01054  0 0 14 01117          61          LDB     PNTR
01055  0 0 53 01046          62          SKN     FLG             SEE IF INT 024 YET
01056  0 0 01 01051          63          BRU     WAIT+2          NO
01057  0 0 10 00000          64          NOP                     YES
01060  0 0 03 01126          65          BRM     SW6TST          TEST SW6
01061  0 0 41 01047          66          BRR     WAIT
                             67 *
                             68 *        INTERRUPT 024 ROUTINE
                             69 *
01062  0 0 00 00000          70 GO       PZE     0
01063  0 0 73 01046          71          SKR     FLG             SET FLG NEG
01064  0 0 10 00000          72          NOP
01065  1 0 57 01062          73          BRC     *GO             CLEAR INTRPT + RET
                             74 *
                             75 *        SETUP ROUTINE TO XMIT DATA
                             76 *
```

```
01066  0 0 00 00000          77 SETUP   PZE    0
01067  0 0 71 01066          78         MPB    SETUP
01070  1 0 16 01066          79         LDA    *SETUP
01071  0 0 76 01117          80         STA    PNTR            ADRS OF LIST
01072  1 0 16 01117          81         LDA    *PNTR           WORD COUNT
01073  0 0 46 01120          82         SKG    ZERO
01074  0 0 41 01066          83         BRR    SETUP           RETURN IF LE ZERO
01075  0 0 76 01121          84         STA    WC              WORD COUNT
01076  0 0 13 01122          85         MRG    M2              CREATE INDEX REG LOAD
01077  1 40 7 0100           86         COPY   (5,1)           PUT IN X1
01100  0 0 16 01117          87         LDA    PNTR
01101  0 0 13 01123          88         MRG    X1              CREAT INDRCT INDEXED
01102  0 0 76 01117          89         STA    PNTR              REF
01103  1 0 26 01117          90         LDP    *PNTR           LOAD 2 WORDS OF LIST
01104  0 1 75 60000          91         STD    BUFFER,1        STORE IN BUFFER
01105  0 1 57 01103          92         BRX    $-2,1     LOOP UNTIL DONE
01106  0 2 02 76702          93         EOM2    2         DISCONNECT CIU
01107  0 0 31 01124          94         POT    STPIT
01110  0 2 02 76602          95         EOM1    2       OUTPUT WORD COUNT
01111  0 0 31 01121          96         POT    WC
01112  0 2 02 76702          97         EOM2   2     OUTPUT CNTRL + ADRS
01113  0 0 31 01125          98         POT    BUFADR
01114  0 0 03 01047          99         BRM    WAIT            WAIT FOR INTRPT
01115  0 0 10 00000         100         NOP
01116  0 0 41 01066         101         BRR    SETUP           RETURN
01117  0 0 00 00000         102 PNTR    PZE    0
01120  0 0 00 00000         103 ZERO    PZE    0
01121  0 0 00 00000         104 WC      PZE    0

01122  1 3 76 00000         105 M2      STA    *0,3            INDX INCR>-2
01123  0 1 00 00000         106 X1      HLT    0,1             INDX REG 1
01124  1 0 00 00000         107 STPIT   HLT    *0              DISC CIU POTWORD
01125  0 2 00 60001         108 BUFADR  HLT    BUFFER+1,2      CIU ADRS POTWORD
                            109 *
                            110 *       SWITCH 6 TEST
                            111 *
01126  0 0 00 00000         112 SW6TST  PZE    0
01127  0 224 0001           113         SNT    001             LOOP IF
01130  0 0 01 01127         114         BRU    $-1               SWITCH 6 ON
```

```
01131  0 0 10 00000          115          NOP
01132  0 0 41 01126          116          BRR    SW6TST
                             117 *
                             118 *        SET CONSOLE INTRPT 033
                             119 *
00033                        120          AORG   033
00033  0 0 01 01000          121          BRU    STRT
                             122 *
                             123 *        SET FIVE BUFFERS EMPTY
                             124 *
02100                        125          AORG   02100
02100  0 0 00 00000          126 BUF1     PZE    0
02200                        127          AORG   02200
02200  0 0 00 00000          128 BUF2     PZE    0
02300                        129          AORG   02300
02300  0 0 00 00000          130 BUF3     PZE    0
02400                        131          AORG   02400
02400  0 0 00 00000          132 BUF4     PZE    0
02500                        133          AORG   02500
02500  0 0 00 00000          134 BUF5     PZE    0
                             135 *
                             136 *        OUTPUT BUFFER AREA
                             137 *
60000                        138          AORG   060000
60000                        139 BUFFER   RES    0100
                             140 *
                             141 *        AN AD/4 REGISTER READ/WRITE ROUTINE
                             142 *
02100                        143          AORG   02100
02100  0 0 00 00004          144          PZE    END1-$          WORD COUNT
02101  0 0 00 01022          145          PZE    01022
02102  0 0 00 06223          146          PZE    06223
02103  0 0 00 01426          147          PZE    01426
02104  25252525              148 END1     DATA   025252525
                             149 *
                             150 *        AN AD/4 ADC READ ROUTINE
                             151 *
02200                        152          AORG   02200
02200  0 0 00 00013          153          PZE    END2-$
02201  0 0 00 01022          154          PZE    01022
02202  0 0 00 06220          155          PZE    06220
02203  0 0 00 02444          156          PZE    02444
02204  0 0 00 01022          157          PZE    01022
```

```
                                                                    DATE

02205  0 0 00 06220          158          PZE     06220
02206  0 0 00 03447          159          PZE     03447
02207  25252525              160          DATA    025252525,025252525,025252525,025252525
02210  25252525
02211  25252525
02212  25252525
02213  25252525              161 END2     DATA    025252525
       00001000              162          END     STRT
```

APPENDIX B

AD/4 - SDS/9300 DISCRETES CHECK OUT PROGRAM

BOK
DYNAMICS, INC.

AD/4 - SDS/9300 DISCRETES CHECK OUT PROGRAM

PURPOSE

This program writes a 16 bit word into the AD/4 control register 0 and reads back a 16 bit word from the AD/4 sense line register 0. This is accomplished by the SDS/9300 via the direct memory access ports and the remote hybrid interface. The word written and the word read back to the SDS/9300 should be equal if the AD/4 logic board is patched so that DGC 1 goes to DGS 1, DGC 2 to DGS 2, etc.

Execution of the program automatically results in a test of all bit patterns between 0_8 and 77776_8. If an error is detected the 9300's B-register display will blink off-and-on 25 times and the next bit pattern will be tested. In addition to blinking the B-register an error message is printed on the TTY. To examine the error condition the program must be stopped and the contents of location 60005 (what was written) compared to location 60015 (what was read back).

USAGE

The program has three options, all of which are accessed by control panel sense switches. These options provide an unconditional program pause, a pause if error is detected and a bypass of TTY output. Specifically the options and their usage are as follows:

SS3	ON	Pause if read $\neq$ write (pauses at 60130)
	OFF	Continue
SS5	ON	Bypass TTY error message
	OFF	Print TTY error message
SS6	ON	Program unconditional pause (pauses in a loop at 60001 to 60002)
	OFF	Continue

To load the "binary deck" the user should follow these steps. First, put binary deck on back of Utility Library Program and load card reader. Second, on the computer console;

1. Press Idle.
2. Press Reset (then Press Load on card reader).
3. Press Clear Flags and Clear together.
4. Press Reset.
5. Press Run.
6. Press SS4.
7. Press Cards.

To execute the program do the following:

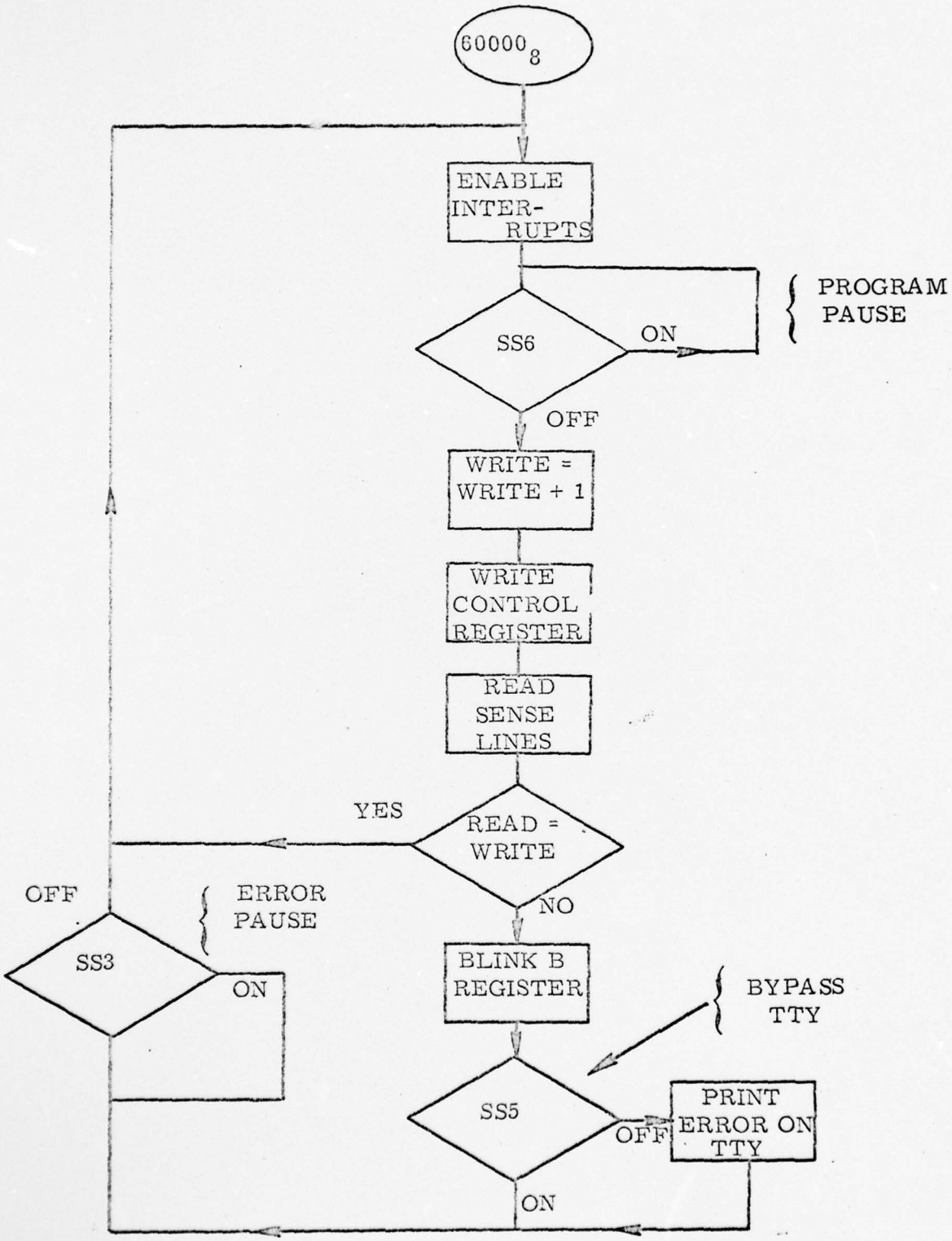
1. Press Idle.
2. Press Reset.
3. Enter BRU 60000_8 in the accumulator.
4. Press Run.

To use the Utility Library Program for displaying data locations do the following:

1. Press Idle.
2. Press Reset.
3. Press Run.
4. Press INT32.
5. On the keyboard enter SNAP___XXXXX, where XXXXXis the location to be displayed.

To re-enter the test program at any time after using the Utility Library Program repeat the steps required for program execution (as shown above).

AD/4 - SDS/9300 DISCRETES CHECK OUT PROGRAM



```
ΔMETA9300 SI,L0,B0                                                    RTM
                                    1 * WRITE AD4 CONTRL REG 0 AND READ BACK AD4 SENSE LINE REG 0
    60000                           2 START  ABRG    060000
    60000  00220002                 3        EIR                   ENABLE INTERUPTS
    60001  0 224 0001                4        SWT     01            USE SSW 6 TO PAUSE
    60002  0 0 01 60001              5        BRU     $-1           PAUSE UNTILL SSW 6 IS OFF
    60003  0 0 10 00000              6        NOP
                                    7 ********************************************************************************
    60004  0 0 71 70003              8 INCR   MPO     070003        INCREMENT WRITE BY ONE
    60005  0 0 71 70005              9        MPO     070005
    60006  0 0 16 70005             10        LDA     070005
    60007  0 0 45 66006             11        SKE     MAX           IF WRITE GT MAX SET TO ZERO
    60010  0 0 01 60016             12        BRU     REWR          MAX WRITE HAS NOT BEEN REACHED
    60011  0 0 16 70016             13        LDA     =000000       MAX WRITE HAS BEEN REACHED
    60012  0 0 76 70003             14        STA     070003        SET WRITE EQ ZERO
    60013  0 0 76 70005             15        STA     070005        SET WRITE EQ ZERO
    60014  0 0 10 00000             16        NOP
    60015  0 0 10 00000             17        NOP
                                    18 ********************************************************************************
    60016  0 0 16 66004             19 REWR   LDA     INT1          PREPARE INTERUPT RETURN
    60017  0 0 76 00024             20        STA     024
    60020  20276602                 21        DATA    020276602     EOM
    60021  0 0 31 66000             22        POT     WCW
    60022  20276702                 23        DATA    020276702     EOM
    60023  0 0 31 66001             24        POT     SADDW
    60024  0 0 01 60024             25        BRU     $             WAIT LOOP FOR AD4
    60025  0 0 57 60026             26 ENDW   BRC     $+1           CLEAR INTERUPT
    60026  0 0 16 66005             27        LDA     INT2          PREPARE INTERUPT RETURN
    60027  0 0 76 00024             28        STA     024
    60030  20276602                 29        DATA    020276602     EOM
    60031  0 0 31 66002             30        POT     WCR
    60032  20276702                 31        DATA    020276702     EOM
    60033  0 0 31 66003             32        POT     SADDR
    60034  0 0 01 60034             33        BRU     $             WAIT LOOP FOR AD4
    60035  0 0 57 60036             34 ENDR   BRC     $+1           CLEAR INTERUPT
```

```
                                    35 ********************************************************************************
60036  0 0 16 70005                 36          LDA     WRITEB+5        TEST FOR WRITE EQUAL READ
60037  0 0 45 70015                 37          SKE     READB+5
60040  0 0 01 60044                 38          BRU     PRINT           READ NOT EQUAL TO WRITE
60041  0 0 01 60000                 39          BRU     START           READ EQUALS WRITE
60042  0 0 10 00000                 40          NOP
60043  0 0 10 00000                 41          NOP
                                    42 ********************************************************************************
60044  0 224 0002                   43 PRINT    SWT     02              USE SSW 5 TO STOP BLINKING LIGHTS
60045  0 0 01 60067                 44          BRU     BLINK           BLINK B REGISTER
60046  0 0 01 60047                 45          BRU     TYPE            TYPE ERROR MESSAGE AND BLIK B REG
60047  0 02 1 02641                 46 TYPE     TYP     *0,1,4
60050  0 02 0 14240                 47          EOM     014240
60051  0 0 31 60054                 48          POT     AMSG
60052  0 20 14000                   49          CAT     0
60053  0 0 01 60052                 50          BRU     $-1
60054  0 0 05 20055                 51 AMSG     ADD     MSG-040000
60055  52255151                     52 MSG      TEXT    40> ERROR IN AD4 DISCRETES
60056  46516031
60057  45602124
60060  04602431
60061  62235125
60062  63256260
60063  60606060
60064  60606060
60065  60606060
60066  60606060
                                    53 ********************************************************************************
60067  0 0 16 70016                 54 BLINK    LDA     =00000          INITIALIZE LOOP COUNTER
60070  0 0 76 66007                 55          STA     LOOPS
60071  0 0 14 70017                 56 LOOP     LDB     =077777         BLINK B REGISTER ON
60072  0 0 16 70016                 57          LDA     =000000
60073  0 0 76 66010                 58          STA     LONS
60074  0 0 16 70020                 59          LDA     =07777
60075  0 0 10 00000                 60 LON      NOP
60076  0 0 10 00000                 61          NOP
60077  0 0 10 00000                 62          NOP
60100  0 0 10 00000                 63          NOP
60101  0 0 10 00000                 64          NOP
```

```
60102  0 0 71 66010          65          MPO     LONS
60103  0 0 45 66010          66          SKE     LONS
60104  0 0 01 60075          67          BRU     LON
60105  0 0 14 70016          68          LDB     =000000         BLINK B REGISTER OFF
60106  0 0 16 70016          69          LDA     =000000
60107  0 0 76 66011          70          STA     LOFFS
60110  0 0 16 70020          71          LDA     =07777
60111  0 0 10 00000          72 LOFF     NOP
60112  0 0 10 00000          73          NOP
60113  0 0 10 00000          74          NOP
60114  0 0 10 00000          75          NOP
60115  0 0 10 00000          76          NOP
60116  0 0 71 66011          77          MPO     LOFFS
60117  0 0 45 66011          78          SKE     LOFFS
60120  0 0 01 60111          79          BRU     LOFF
60121  0 0 71 66007          80          MPO     LOOPS
60122  0 0 16 70021          81          LDA     =025            MAXIMUM COUNT
60123  0 0 45 66007          82          SKE     LOOPS
60124  0 0 01 60071          83          BRU     LOOP            BLINK LOOP NOT FINISHED
60125  0 0 10 00000          84          NOP
60126  0 0 10 00000          85          NOP
                             86 ********************************************************************************
60127  0 0 10 00000          87          NOP
60130  0 224 0010            88          SWT     010             USE SSW 3 FOR AUTO STOP AT ERROR
60131  0 0 01 60130          89          BRU     $-1
60132  0 0 10 00000          90          NOP
60133  0 0 01 60000          91          BRU     START           BLINK LOOP IS FINISHED
                             92 ********************************************************************************
66000                        93          AORG    066000          DATA FOR PROGRAM
66000  00000006              94 WCW      DATA    06              WORD COUNT FOR WRITE
66001  20070000              95 SADDW    DATA    020070000       STARTING ADDRESS FOR WRITE
66002  00000006              96 WCR      DATA    06              WORD COUNT FOR READ
66003  20070010              97 SADDR    DATA    020070010       STARTING ADDRESS FOR READ
66004  0 0 01 60025          98 INT1     BRU     ENDW            INTERUPT RETURN FROM TRAP TO 24
66005  0 0 01 60035          99 INT2     BRU     ENDR            INTERUPT RETURN FROM TRAP TO 24
66006  00077776             100 MAX      DATA    077776          MAXIMUM VALUE OF CONTROL WORD
66007                       101 LOOPS    RES     01              CORRENT LOOP COUNT FOR BLINK LIGHT
66010                       102 LONS     RES     01              CYCLES OF LIGHT OFF
66011                       103 LOFFS    RES     01              CYCLES OF LIGHT OFF
                            104 ********************************************************************************
```

```
70000                          105 WRITEB  AORG    0/0000      DATA FOR DMA WRITE
70000  00001021                106         DATA    01021       WRITE IN IRC
70001  00000020                107         DATA    020         ANALOG RUN MODE
70002  00001050                108         DATA    01050       WRITE INTO CLR
70003  00000000                109         DATA    0000000     FIRST WORD WRITTEN IN CLR
70004  00001050                110         DATA    01050       WRITE INTO CLR
70005  00000000                111         DATA    0000000     SECOND WORD WRITTEN INTO CLR
                               112 **********************************************************
70010                          113 READB   AORG    0/0010      DATA FOR DMA READ
70010  00001021                114         DATA    01021       WRITE INTO IRC
70011  00000020                115         DATA    020         ANALOG RUN MODE
70012  00001460                116         DATA    01460       READ SENSE LINES
70013                          117         RES     01          FIRST WORD READ ON SENSE LINES
70014  00001460                118         DATA    01460       READ SENSE LINES
70015                          119         RES     01          SECOND WORD READ ON SENSE LINES
                               120         END

70016  00000000
70017  00077777
70020  00007777
70021  00000025
```

APPENDIX C

3010 DIAGNOSTIC PROGRAM

BOK
DYNAMICS, INC.

3010 DIAGNOSTIC PROGRAM

PURPOSE

The program writes and reads back blocks of 23 words from locations $FEOO_{16}$ to $FFCO_{16}$ in 3010 core. Each pass the pattern being transmitted is incremented by one. The pattern is incremented from zero to 177777_8 and reset to zero when the high count is reached.

If an error is detected in a block the contents of the 9300 read buffer is printed along with data pattern that was transmitted (see example). To suppress the compare function set sense switch 1.

Every $10,000_8$ passes through the blocks $FEOO_8$ to $FFCO_8$ a print is made (see example) to indicate that the program is operating. (This may be effectively suppressed by loading $7777\ 7777_8$ into location STOPC (60130_8).

USAGE

To load the program into memory do the following:

1. Put BKD program loader card in hopper.
2. Ready card reader.
3. On the console, push Idle then Reset.
4. Clear register lights.
5. Hold down Clear and Clear Flags for 1 second.
6. Press Reset, Run, and Cards.
7. Program counter will stop at _____ indicating load completed.

To execute the program which has been previously loaded do the following:

1. Press Idle, Reset.
2. With register display set to B, enter 00160200_8.
3. Clear the 3010 interface.
4. Press Run.

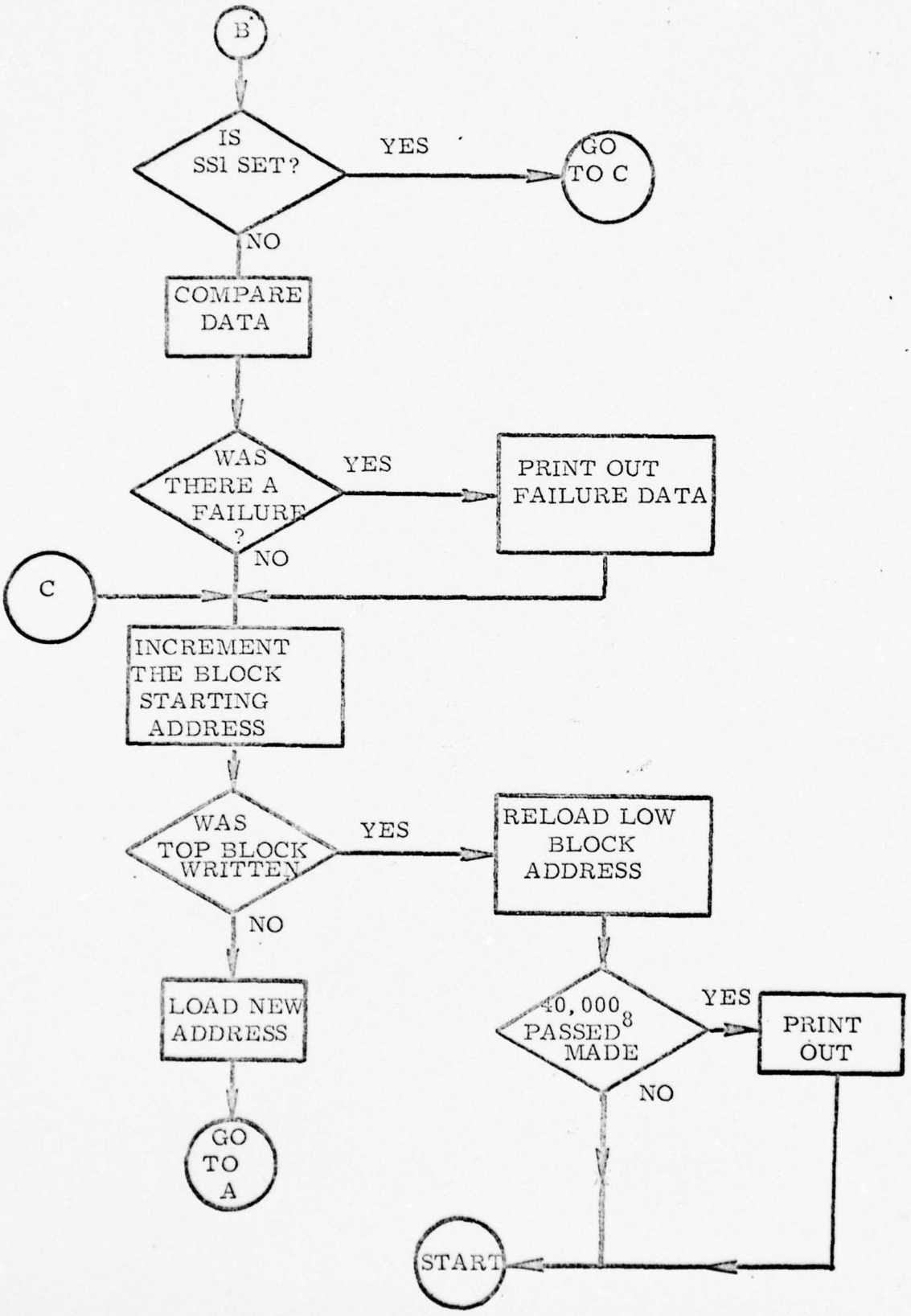
To suppress data comparison set sense switch 1. To halt set sense switch 6. To modify block length the following changes must be made.

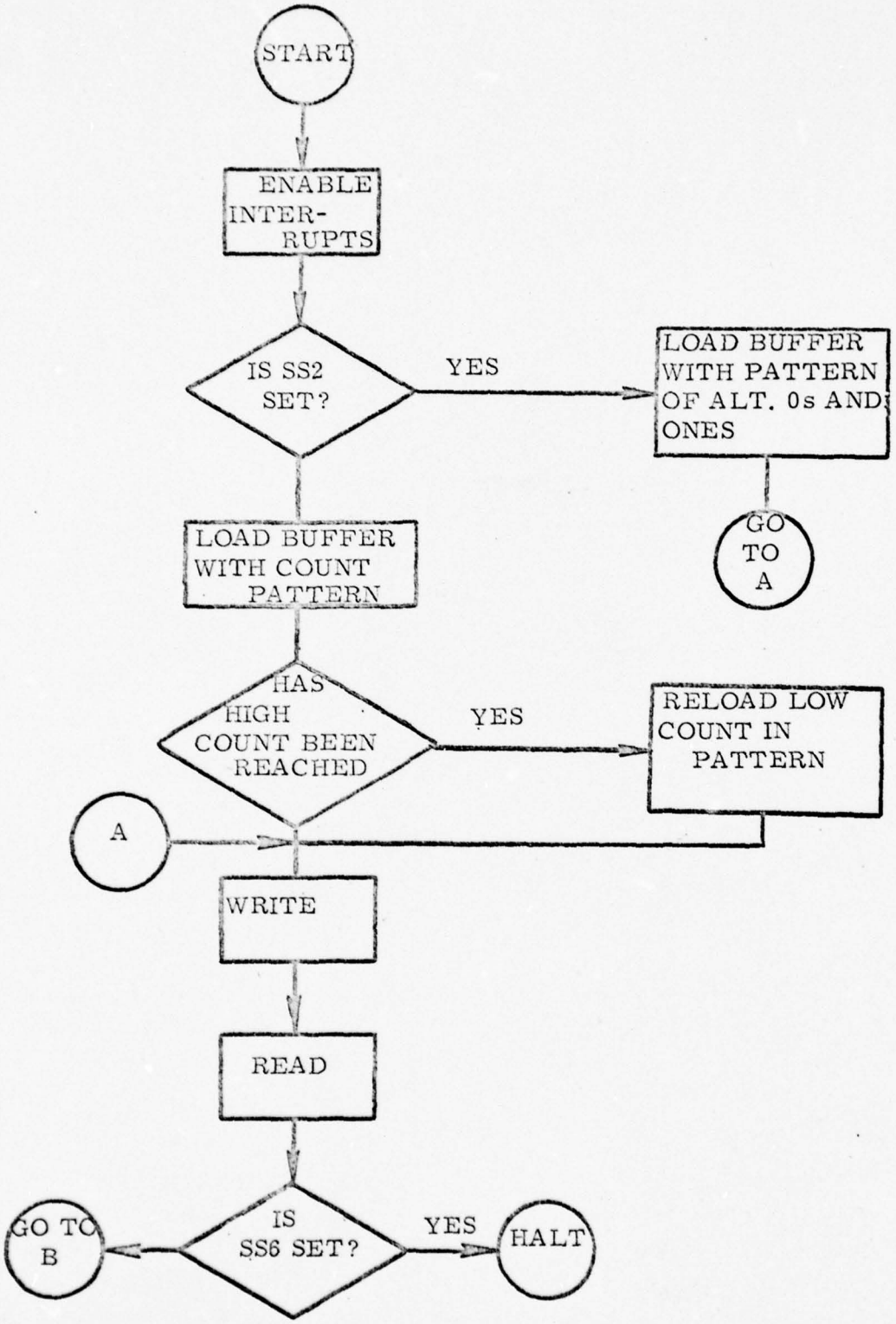
1. LASTLOC (loc. 60117_8) must contain 60040_8 + No. words in the block to be transmitted.
2. WC1 (loc. 60112_8) and WC2 (loc. 60113_8) must contain (No. words transmitted + 1)$_8$.
3. INCR (loc. 60114) must contain (2 X No. Words)$_8$ in the block to be transmitted. For example - to do a 4 word transfer, enter:

WC1	60112	00000005	($4_8 + 1_8$)
WCZ	60113	00000005	
INCR	60114	00000010	(2_8 X 4_8 = 10)
LASTLOC	60117	00060044	($60040_8 + 4_8$)

To change the pattern low count load LCNT (60105) with XXXXXXXX where XXXXXXXX is the desired low count. Note the program normally counts from zero to 00177777.

SDS/9300 to GE/3010 INTERFACE CHECKOUT PROGRAM




START
ENABLE INTER-RUPTS
IS SS2 SET?
YES
LOAD BUFFER WITH PATTERN OF ALT. 0s AND ONES
GO TO A
LOAD BUFFER WITH COUNT PATTERN
HAS HIGH COUNT BEEN REACHED
YES
RELOAD LOW COUNT IN PATTERN
A
WRITE
READ
IS SS6 SET?
YES
HALT
GO TO B

```
                                    1 *3010/9300 INTERFACE DIAGNOSTIC PROGRAM
60200                               2          ORG     060200
60200  00220002                     3 START    EIR
60201  0 224 0020                   4          SWT     020             SET SS 2 TO WRITE ALT ONES AND ZEROES
60202  0 0 01 60327                 5          BRU     PATSEQ
                                    6 * LOAD PATTERN IN WRITE BUFFER (STARTS AT 60001)
60203  0 0 16 60121                 7 LOAD     LDA     FADD
60204  0 0 76 60122                 8          STA     TEMPA
60205  0 0 16 60102                 9          LDA     PATTRN
60206  0 0 76 60123                10          STA     TEMPP
60207  1 0 76 60122                11 BACK    STA     *TEMPA
60210  0 0 71 60122                12          MPO     TEMPA
60211  0 224 0020                  13          SWT     020
60212  0 0 01 60340                14          BRU     ZAO
60213  0 0 16 60122                15 ZBACK   LDA     TEMPA
60214  0 0 46 60124                16          SKG     LADD
60215  0 0 01 60217                17          BRU     CYCLE
60216  0 0 01 60221                18          BRU     PCHK
60217  0 0 16 60123                19 CYCLE   LDA     TEMPP
60220  0 0 01 60207                20          BRU     BACK
60221  0 0 71 60102                21 PCHK    MPO     PATTRN
60222  0 0 16 60102                22          LDA     PATTRN
60223  0 0 46 60103                23          SKG     HCNT            HIGH COUNT REACHED
60224  0 0 01 60240                24          BRU     WRITE           GO TO WRITE / READ
60225  0 0 16 60104                25          LDA     LCNT            RESET COUNTER TO LOW COUNT
60226  0 0 76 60102                26          STA     PATTRN
60227  0 0 01 60240                27          BRU     WRITE
                                   28 *WRITE/READ MEMORY
60240                              29          ORG     060240
60240  0 0 16 60105                30 WRITE   LDA     RET1
60241  0 0 76 00020                31          STA     020
60242  20276604                    32          DATA    00202766Q4
60243  0 0 31 60111                33          PBT     WC1
60244  20276704                    34          DATA    00202767Q4
60245  0 0 31 60107                35          PBT     SA1
60246  0 0 01 60246                36          BRU     $
60247  0 0 16 60106                37          LDA     RET2
60250  0 0 76 00020                38          STA     020
60251  20276604                    39          DATA    00202766Q4
60252  0 0 31 60112                40          PBT     WC2
60253  20276704                    41          DATA    00202767Q4
60254  0 0 31 60110                42          PBT     SA2
```

```
60255  0 0 01 60255        43          BRU     $
60256  0 0 10 00000        44          NOP
60257  0 224 0001          45          SWT     001             SET SS6 TO HALT
60260  0 0 00 00000        46          HLT
60261  0 0 01 60271        47          BRU     INCPT
                           48 * INCREMENT THE BLOCK STARTING ADDRESS
60262  0 0 16 60000        49 INCRM    LDA     WBA
60263  0 0 05 60113        50          ADD     INCR
60264  0 0 44 60117        51          SKL     MTOP

60265  0 0 01 60310        52          BRU     ARELD           GO TO MEMORY ADDR RELOAD ROUTINE
60266  0 0 76 60000        53          STA     WBA
60267  0 0 76 60040        54          STA     RBA
60270  0 0 01 60240        55          BRU     WRITE
60271  0 224 0040          56 INCPT    SWT     040             SET SS1 TO SUPPRESS COMPARE AND PRINT
60272  0 0 01 60262        57          BRU     INCRM
                           58 * COMPARE DATA WRITTEN WITH DATA READ
60273  0 0 16 60114        59 CMPAR    LDA     WSTA
60274  0 0 76 60100        60          STA     TWSTA
60275  0 0 16 60115        61          LDA     RSTA
60276  0 0 76 60101        62          STA     TRSTA
60277  1 0 16 60100        63 CAGN     LDA     *TWSTA
60300  1 0 45 60101        64          SKE     *TRSTA
60301  0 0 01 60323        65          BRU     PRINT
60302  0 0 71 60100        66          MPO     TWSTA
60303  0 0 71 60101        67          MPO     TRSTA
60304  0 0 16 60101        68          LDA     TRSTA
60305  0 0 45 60116        69          SKE     LSTLOC          QUIT IF LAST LOCATION
60306  0 0 01 60277        70          BRU     CAGN
60307  0 0 01 60262        71          BRU     INCRM
                           72 * MEMORY BLOCK ADDRESS RELOAD
60310  0 0 16 60120        73 ARELD    LDA     LBSA
60311  0 0 76 60000        74          STA     WBA
60312  0 0 76 60040        75          STA     RBA
60313  0 0 71 60127        76          MPO     PCNTR
60314  0 0 16 60127        77          LDA     PCNTR
60315  0 0 45 60130        78          SKE     STOPC
60316  0 0 01 60200        79          BRU     START
60317  0 0 03 02106        80          BRM     02106
60320  0 0 77 60127        81          STZ     PCNTR
```

```
60321  0 0 71 60127      82          MPO     PCNTR
60322  0 0 01 60200      83          BRU     START
                         84 * ENTRY TO UTILITY PRINT ROUTINE
60323  0 0 76 60071      85 PRINT    STA     RBA+25
60324  0 0 03 02106      86          BRM     02106
60325  0 0 77 60071      87          STZ     RBA+25
60326  0 0 01 60262      88          BRU     INCRM
                         89 *SET INITIAL PATTRN TO ZERO OR ONE
60327  0 0 16 60133      90 PATSEQ   LDA     PATRNT
60330  0 0 76 60102      91          STA     PATTRN
60331  0 0 45 60131      92          SKE     ZEROS
60332  0 0 01 60336      93          BRU     X
60333  0 0 16 60132      94          LDA     ONES
60334  0 0 76 60133      95          STA     PATRNT
60335  0 0 01 60203      96          BRU     LOAD
60336  0 0 77 60133      97 X        STZ     PATRNT
60337  0 0 01 60203      98          BRU     LOAD
                         99 * ALTERNATE THE PATTRN FROM ONES TO ZEROES
60340  0 0 16 60123     100 ZAO      LDA     TEMPP
60341  0 0 45 60132     101          SKE     ONES
60342  0 0 01 60345     102          BRU     Z
60343  0 0 77 60123     103          STZ     TEMPP
60344  0 0 01 60213     104          BRU     ZOBAC

60345  0 0 16 60132     105 Z        LDA     ONES
60346  0 0 76 60123     106          STA     TEMPP
60347  0 0 01 60213     107          BRU     ZOBAC
60000                   108          ABRG    060000
                        109 * DATA BUFFER WRITE
60000  00177000         110 WBA      DATA    0177000
60001                   111          RES     035
                        112
                        113 * DATA BUFFER READ
60040                   114          ABRG    060040
60040  00177000         115 RBA      DATA    0177000
60041                   116          RES     035
60076  00060000         117 WSA      DATA    060000
60077  00060040         118 RSA      DATA    060040
```

```
60100  0 0 00 00000          119 TMSTA   PZE
60101  0 0 00 00000          120 TRSTA   PZE
60102  00000001              121 PATTRN  DATA    01
60103  00177777              122 HCNT    DATA    000177777
60104  00000000              123 LCNT    DATA    000000000
60105  00760247              124 RET1    DATA    000760247
60106  00760256              125 RET2    DATA    000760256
60107  20060000              126 SA1     DATA    020060000
60110  20060040              127 SA2     DATA    020060040
60111  00000030              128 WC1     DATA    030
60112  00000030              129 WC2     DATA    030
60113  00000056              130 INCR    DATA    056
60114  00060001              131 WSTA    DATA    060001
60115  00060041              132 RSTA    DATA    060041
60116  00060067              133 LSTLOC  DATA    060067
60117  00177700              134 MTOP    DATA    0177700
60120  00177000              135 LBSA    DATA    0177000
60121  00060001              136 FADD    DATA    060001
60122  0 0 00 00000          137 TEMPA   PZE
60123  0 0 00 00000          138 TEMPP   PZE
60124  00060031              139 LADD    DATA    060031
60125  0 0 00 00000          140 TBSAW   PZE
60126  0 0 00 00000          141 TBSAR   PZE
60127  0 0 00 00000          142 PCNTR   PZE
60130  00040000              143 STOPC   DATA    040000
60131  00000000              144 ZEROS   DATA    000000000
60132  00177777              145 ONES    DATA    000177777
60133  00177777              146 PATRNT  DATA    000177777
                             147         END
```

APPENDIX D

FORTRAN HYBRID SOFTWARE

BOK
DYNAMICS, INC.

FORTRAN HYBRID SOFTWARE

This Appendix contains the real-time software for the STINGER simulation. The code presented here was converted from IBM-7094 MAP into CDC-6600 FORTRAN hybrid. The statements denoted SOFT-T pertain to the software test proceedures. The statements denoted SOFT-MOD correct inconsistencies between the FTN. and FTHH. compilers.

SUBROUTINE FLIGHT

PURPOSE

This program serves as a driver for the STINGER real-time code. The program initializes parameters, reserves DADIOS equipment and transfers initial conditions to the AD-4.

DESCRIPTION OF PARAMETERS

(see code)

SUBROUTINES REQUIRED

- SUBROUTINE ADFOUR
- SUBROUTINE SIMRUN
- SUBROUTINE REALT
- SUBROUTINE BHOLD
- SUBROUTINE RES

```
      SUBROUTINE FLIGHT
C
C     PROGRAM VARIABLES
C        MAXBIT          MAXIMUM BITS CONVERTED IN INPUT SENSE LINE
C        BIT(I)          BIT CONVERSION OF INPUT SENSE LINE
C        IIN             INPUT SENSE LINE
C        IOUT            OUTPUT SENSE LINE
C                        SENSE LINE 0 = 0000000000000001 =   1
C                        SENSE LINE 1 = 0000000000000010 =   2
C                        SENSE LINE 2 = 0000000000000100 =   4
C                        SENSE LINE 3 = 0000000000001000 =   8
C                        SENSE LINE 4 = 0000000000010000 =  16
C                        SENSE LINE 5 = 0000000000100000 =  32
C                        SENSE LINE 6 = 0000000001000000 =  64
C                        SENSE LINE 7 = 0000000010000000 = 128
C                        SENSE LINE 8 = 0000000100000000 = 256
C        LAUNCH(I)       DAC VARIABLES
C        ADIN(I)         AOC VARIABLES
C        IWRITE          IF IWRITE = 1, WRITE COMMENTS
C
      REAL MAN(200),MISSED(7),MISS,LAUNCH
      INTEGER WMAN
      DIMENSION FIN(10),TS(30)
      EQUIVALENCE (TS(1), TMAS(1)),(MAN(1),XMAN(1)),(MISSED(1),XMISS(1))
      EQUIVALENCE (FIN(1),DX),(FIN(2),DY),(FIN(3),DZ),(FIN(4),DT)
      EQUIVALENCE (FIN(5),XDOT),(FIN(6),YDOT),(FIN(7),ZDOT)
      EQUIVALENCE (FIN(8),XXX),(FIN(9),YYY),(FIN(10),ZZZ)
      COMMON/EXTRA/IT1,KCK,ICR2,IDA3,IND,INDEX
      COMMON/COMA/LEVEL,IPTS,XXS(50),XDTGO,YDTGO,ZDTGO,RLB,COSE,SPO,RI,
     *GAM,EDOT,THETAL,RN,
     *              PPX(50),PPY(50),PPZ(50),TIME(50),TMAS(30),XDTGMS(30)
     *             ,YDTGMS(30),ZDTGMS(30),XMAN(4,50),XMISS(7),NT
     *             ,XCOMP,YCOMP,ZCOMP,TAMA(30),DELTAR(30),VM(30),G,GGG
     *,XDO,YDO,ZDO,DXG,DYG,DZG,S2,S3,S4,S5,XDM(30),YDM(30),ZDM(30),
     *RLBK,SCALEP,F1,F2,F3,G1,G2,G3,XC,YC,ZC,S1,RRR,SR,SPL,CTL,STL,
     *CPL,A1,VTI,XE(30),YE(30),ZE(30),ZALT,NERR,CLA,NPX,NX,CLAA(10),
     *PHO,ARG,AAA,SCALET,TREAL,TMA(30),XTA,YTA,ZTA,SCALEV,QMM(10),QM
     *,SA,CA,VMX(50),VMY(50),VMZ(50)
      COMMON/INTCOM/IBIT(60),MAXBIT,IWRITE
      COMMON/ZADC1/ADIN(10)                                          SOFT-MOD
      COMMON/ZDAC1/LAUNCH(11)                                        SOFT-MOD
      COMMON/ZODIS2/IOUT                                             SOFT-MOD
      COMMON/ZIDIS2/IIN                                              SOFT-MOD
      LEVEL=7
      IT1=1
      KCK=-1
      ICR2=29
      IDA3=0
      IND=0
      INDEX=0
      MAXBIT=16
      IOUT=0
      IWRITE = 1
      DO 3 I=1,MAXBIT
    3 IBIT(I)=0
C
C     WAIT FOR STATIC CHECK COMPLETE ( BIT 5 )
```

```
    1 CONTINUE                                                          SOFT-T
      CALL ADFOUR                                                       SOFT-T
      CALL SIMRUN(ISTAT)
      WRITE(6,2000)ISTAT
      IF(ISTAT.GT.0)STOP
      CALL REMARK(17H JOB IN REAL TIME)
      IF(IBIT(5).NE.1)GO TO 1
      IF(IWRITE.EQ.1)WRITE(6,3000)
C
C     MOVE OUTPUT DATA TO DACS
C
      LAUNCH(1)=XDTGO
      LAUNCH(2)=YDTGO
      LAUNCH(3)=ZDTGO
      LAUNCH(4)=RLB
      LAUNCH(5)=COSE
      LAUNCH(6)=SPO
      LAUNCH(7)=RI
      LAUNCH(8)=GAM
      LAUNCH(9)=EDOT
      LAUNCH(10)=THETAL
      LAUNCH(11)=RN
      WRITE(6,5001)(LAUNCH(II),II=1,11)                                 SOFT-T
 5001 FORMAT(8H LAUNCH=,11F5.2)                                         SOFT-T
C
C     SEND STATUS BIT TO AD/4 INDICATING ICS SENT ( BIT 4 )
C
      IOUT=16
      CALL ADFOUR                                                       SOFT-T
C
C     WAIT FOR RAMP UP SIGNAL FROM AD/4 (BIT 6)
C
    2 CONTINUE
      CALL ADFOUR                                                       SOFT-T
      IF(IBIT(6).NE.1)GO TO 2
      IF(IWRITE.EQ.1)WRITE(6,4000)
C
      CALL REALT                                                        SOFT-T
      CALL SHOLD
 2000 FORMAT(18H REAL TIME STATUS=,02)
 3000 FORMAT(22H STATIC CHECK COMPLETE)
 4000 FORMAT(23H RAMP UP SIGNAL RECIVED)
      RETURN
      END
```

SUBROUTINE REALT

PURPOSE

This program contains the real-time digital computer computations required for the STINGER simulation. Program inputs are received from the AD-4 via DADIOS ADCs and discretes. The program outputs are transmitted to the AD-4 via DADIOS DACs and discretes.

SUBROUTINES REQUIRED

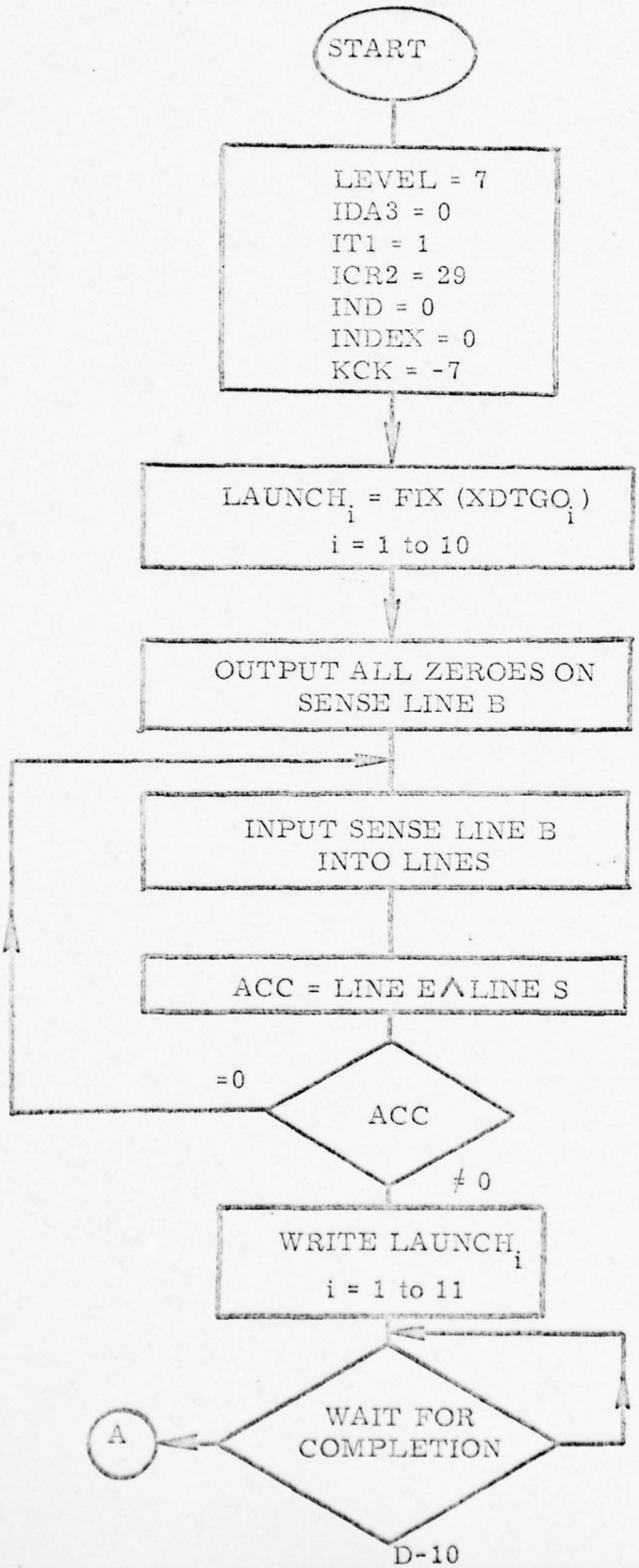
- SUBROUTINE SIMSTOP
- SUBROUTINE SIMHOLD
- SUBROUTINE ADFOUR
- SUBROUTINE SIMIDLE

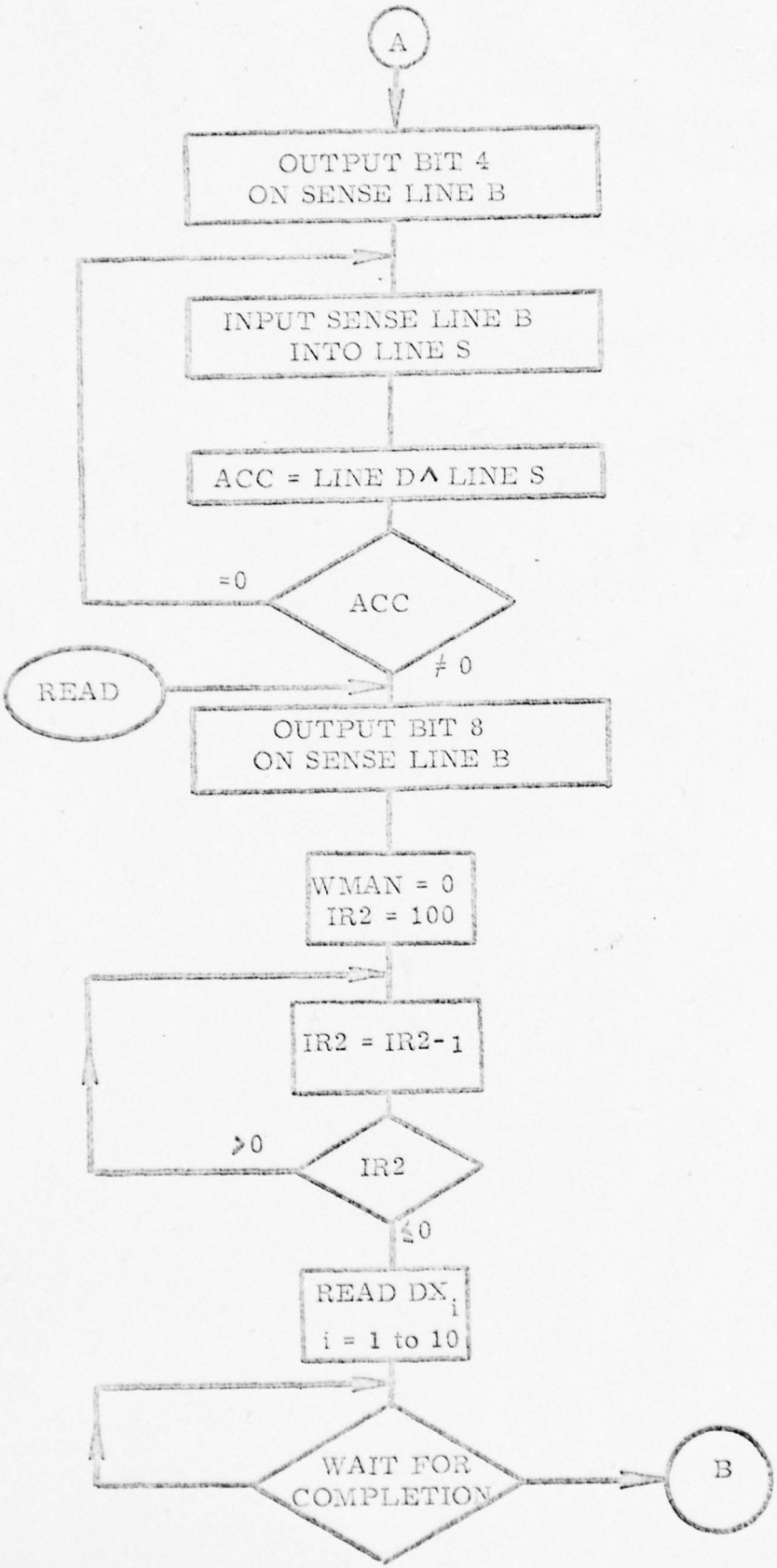
```
      SUBROUTINE REALT
      REAL MAN(200),MISSED(7),MISS,LAUNCH
      INTEGER WMAN
      DIMENSION FIN(10),TS(30)
      EQUIVALENCE (TS(1), TMAS(1)),(MAN(1),XMAN(1)),(MISSED(1),XMISS(1))
      EQUIVALENCE (FIN(1),DX),(FIN(2),DY),(FIN(3),DZ),(FIN(4),DT)
      EQUIVALENCE (FIN(5),XDOT),(FIN(6),YDOT),(FIN(7),ZDOT)
      EQUIVALENCE (FIN(8),XXX),(FIN(9),YYY),(FIN(10),ZZZ)
      COMMON/EXTRA/IT1,KCK,ICR2,IOA3,IND,INDEX
      COMMON/COMA/LEVEL,IPTS,XXS(50),XDTGO,YDTGO,ZDTGO,RL3,COSE,SPO,RI,
     *GAM,EDOT,THETAL,RN,
     *                PPX(50),PPY(50),PPZ(50),TIME(50),TMAS(30),XDTGMS(30)
     *               ,YDTGMS(30),ZDTGMS(30),XMAN(4,50),XMISS(7),NT
     *               ,XCOMP,YCOMP,ZCOMP,TAMA(30),DELTAR(30),VM(30),G,GGG
     *,XDO,YDO,ZDO,DXG,DYG,DZG,S2,S3,S4,S5,XDM(30),YDM(30),ZDM(30),
     *RLBK,SCALEP,F1,F2,F3,G1,G2,G3,XC,YC,ZC,S1,RRR,SR,SPL,CTL,STL,
     *CPL,A1,VTI,XE(30),YE(30),ZE(30),ZALT,NERR,CLA,NPX,NK,CLAA(10),
     *PHO,ARG,AAA,SCALET,TREAL,TMA(30),XTA,YTA,ZTA,SCALEV,QMM(10),QM
     *,SA,CA,VMX(50),VMY(50),VMZ(50)
      COMMON/INTCOM/IBIT(60),MAXBIT,IWRITE
      COMMON/ZADC1/ADIN(10)                                        SOFT-MOD
      COMMON/ZDAC1/LAUNCH(11)                                      SOFT-MOD
      COMMON/ZIDIS2/IIN                                            SOFT-MOD
      COMMON/ZODIS2/IOUT                                           SOFT-MOD
C
C     WAIT FOR FRAME SYNC FROM IRSS ( BIT 8)
 1000 CALL ADFOUR                                                  SOFT-T
      IF(IBIT(8).NE.1)GO TO 1000
      WMAN=0
      MAX=50
C
C     READ ADCS
C
      DO 2 I=1,10
    2 FIN(I)=ADIN(2)
      CALL ADFOUR                                                  SOFT-T
C
C     SCALE ADCS
C
      DO 3 I=1,3
    3 FIN(I)=FIN(I)*SCALEP
      IF(XXS(INDEX+1).GT.DX)GO TO 5
C
C     GO TO 3 CHANNEL MODE
C
C     OUTPUT BIT 3 TO AD/4
C
      ICUT=8
      CALL ADFOUR                                                  SOFT-T
C
      PPX(INDEX+1)=FIN(1)
      PPY(INDEX+1)=FIN(2)
      PPZ(INDEX+1)=FIN(3)
      TIME(INDEX+1)=FIN(4)
      VMX (INDEX+1)=FIN(5)
      VMY (INDEX+1)=FIN(6)
      VMZ (INDEX+1)=FIN(7)
```

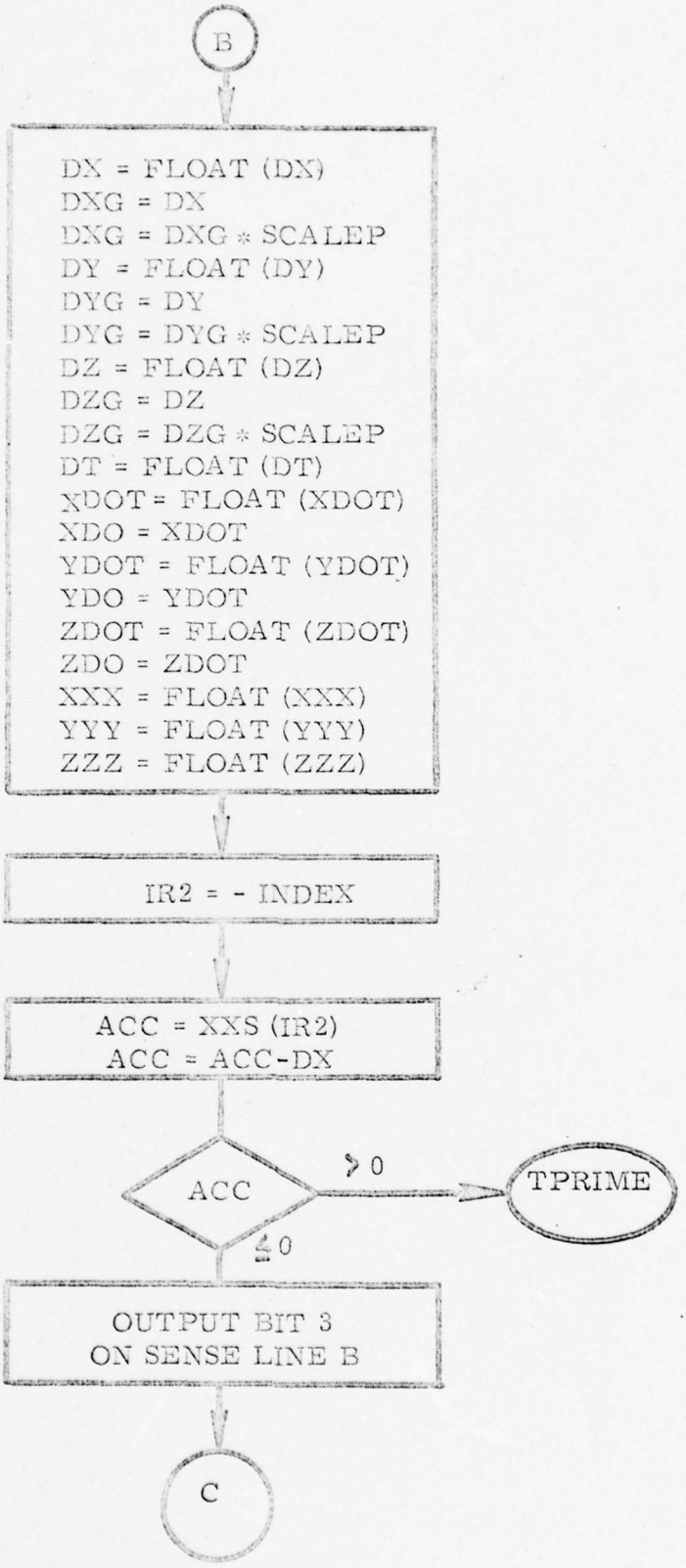
```
      NMAN=7
      IOA3=1
C
C     CKMISS
C
    5 CONTINUE
      IF(DT/10.0.LE.G)GO TO 10
      SKK=GGG+1.0
      KCK=1
      MISS=DX*XDOT+DY*YDOT+DZ*ZDOT
      IF(MISS.LE.0.0)GO TO 10
      MISSED(1)=FIN(1)
      MISSED(2)=FIN(2)
      MISSED(3)=FIN(3)
      MISSED(4)=FIN(4)
      MISSED(5)=FIN(5)
      MISSED(6)=FIN(6)
      MISSED(7)=FIN(7)
  300 CONTINUE
C
C     SYSTEM HOLD (SEND BIT 7 TO AD/4)
      IOUT=128
      CALL ADFOUR                                                    SOFT-T
C
C     RETURN TO BATCH JOB
C
      CALL SIMSTOP
      RETURN                                                         SOFT-T
C     CKTIME
C
   10 CONTINUE
      IF(LEVEL.GT.0)GO TO 1000
      IF(DT.LT.TS(1))GO TO 1000
      LEVEL=0
   20 CONTINUE
      IF(DT.LE.TS(30-ICR2))GO TO 30
      IT1=IT1+1
      IF(IT1.EQ.NT) GO TO 25
      IF(ICR2.LE.1)GO TO 25
      ICR2=ICR2-1
      GO TO 20
   25 CONTINUE
      LEVEL=7
      GO TO 1000
C
   30 CONTINUE
      IT2=IT1+1
      IF(WMAN.EQ.0)GO TO 100
      IND=IND+1
      IF(MAX.LT.IND)GO TO 100
      MAN(4*IND-3)=DT
  100 CONTINUE
C
C     CALC
C
      DIV=TS(IT2)-TS(IT1)
      RATIO=(DT-TS(IT1))/DIV
```

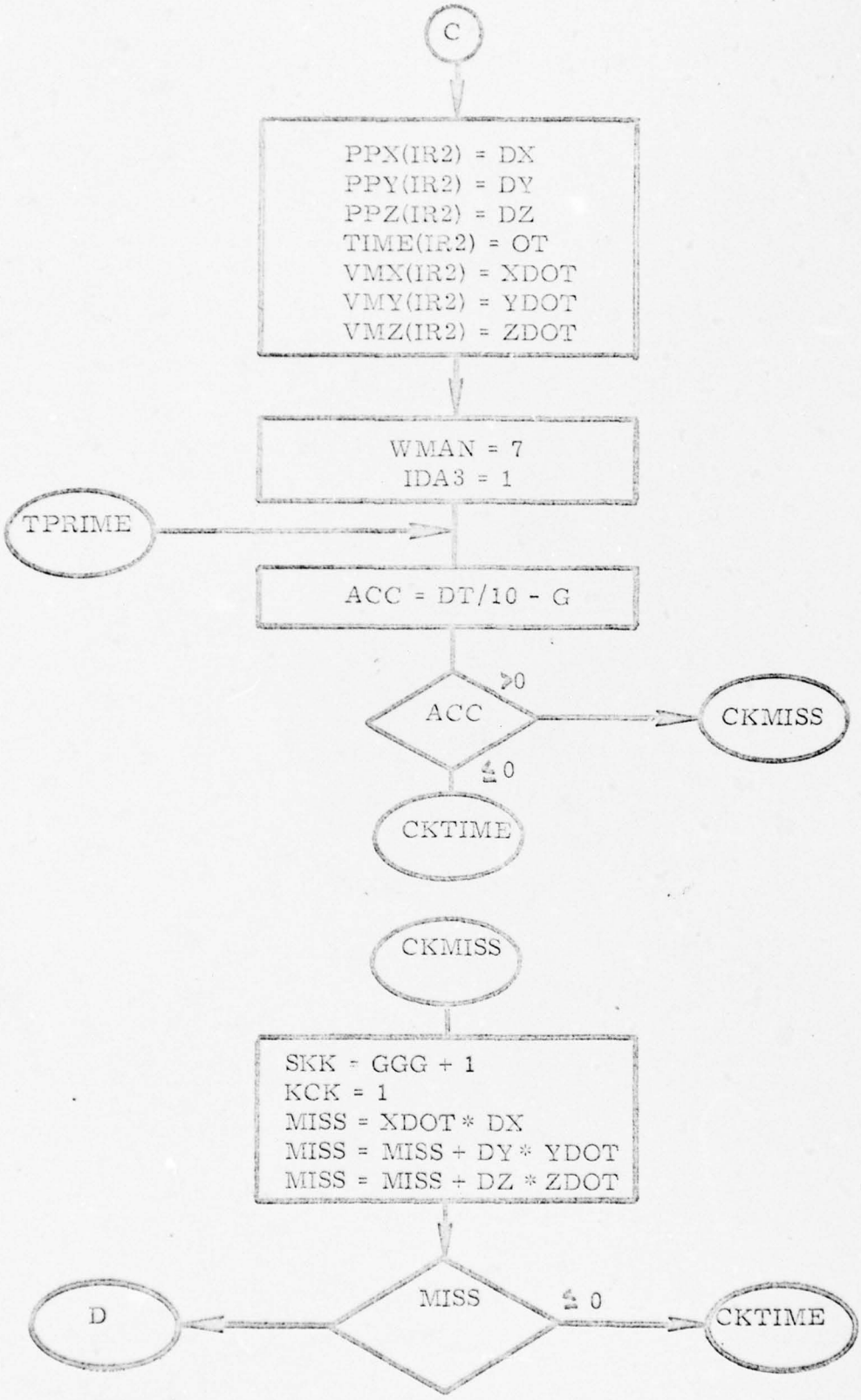
```
      XCOMP=XDTGMS(IT1)+RATIO*(XDTGMS(IT2)-XDTGMS(IT1))
      XC=XDM(IT1)+RATIO*(XDM(IT2)-XDM(IT1))
      YCOMP=YDTGMS(IT1)+RATIO*(YDTGMS(IT2)-YDTGMS(IT1))
      YC=YDM(IT1)+RATIO*(YDM(IT2)-YDM(IT1))
      ZCOMP=ZDTGMS(IT1)+RATIO*(ZDTGMS(IT2)-ZDTGMS(IT1))
      ZC=ZDM(IT1)+RATIO*(ZDM(IT2)-ZDM(IT1))
      TREAL=DT*SCALET
      IF(KCK.LT.0)SSK=1.0+GG3*TREAL/G
      IF(IOA3.GT.0)GO TO 200
      AAA=0.003894*ZALT+1116.89
      VTI=SQRT(XC*XC+YC*YC+ZC*ZC)
      IF(VTI.LT.338.0)GO TO 200
      ARG=0.00003*ZALT
      PHO=0.0023769*EXP(ARG)
      QM=VTI/AAA
      NPX=2
      NX=8
      CALL INTERP(QM,QNM,CLAA,NX,NPX,CLA,NERR)
      CALL INTERP(TREAL,TMA,XE,NT,NPX,XTA,NERR)
      CALL INTERP(TREAL,TMA,YE,NT,NPX,YTA,NERR)
      CALL INTERP(TREAL,TMA,ZE,NT,NPX,ZTA,NERR)
      A1=XTA*XTA+YTA*YTA+ZTA*ZTA
      A1=0.01745329*4.637084242*SQRT(A1)/(PHO*VTI*VTI*CLA)
      SA=SIN(A1)
      CA=COS(A1)
      XXX=XXX*20475.0/SKK
      YYY=YYY*20475.0/SKK
      ZZZ=ZZZ*20475.0/SKK
      RRR=XXX*XXX+YYY*YYY+ZZZ*ZZZ
      RRR=VTI*SQRT(RRR)
      F1=S2*XXX-YYY*SPL+S3*ZZZ
      F2=XXX*S4+YYY*CPL+ZZZ*S5
      F3=CTL*ZZZ-STL*XXX
      SR=SQRT(XC*XC+YC*YC)
      G1=CA+S4/SR*ZC
      G1=S1*XC
      G2=S1*YC
      G3=ZC*CA-SA*SR
      E111=(F1*G1+F2*G2+F3*G3)/RRR
      COSE=1.0-E111*E111
      RLB=SQRT(COSE)
      RLB=RLB*RLBK/SCALET
      COSE=E111/1.02375
      RC1=SQRT(F1*F1+F2*F2)
      RC1=F2/RC1
      RCB=ACOS(RC1)
      RCX=-RC1
      RCY=-SIN(RCB)
      IF(F1.GE.0.0)RCY=-RCY
      F11=(F2*G3-F3*G2)/VTI
      F22=(G1*F3-G3*F1)/VTI
      F33=(G2*F1-G1*F2)/VTI
      FCR=SQRT(F11*F11+F22*F22+F33*F33)
      G11=F11*RCX
      G22=F22*RCY
      C111=(G11+G22)/FCR
      T111=ACOS(C111)
```

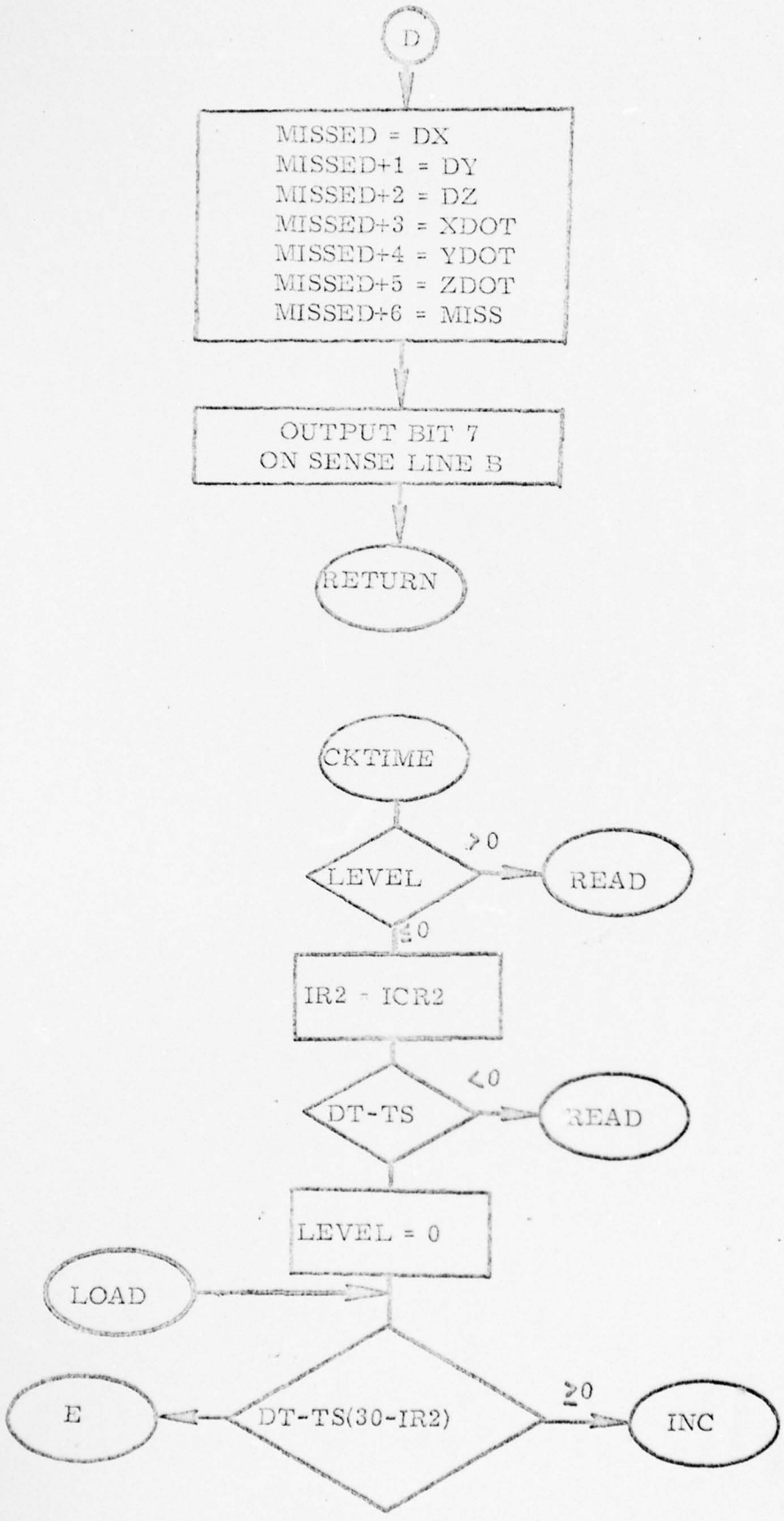
```
      IF(F33.GE.0.0)GO TO 155
      IF(T111.LT.1.570796326)GO TO 156
      TRP=1.570796326+T111
      GO TO 159
  155 TRP=1.570796326-T111
      GO TO 159
  156 TRP=T111-4.712388980
  159 SPO=TRP/SCALET
  200 CONTINUE
C
C     HERE
C
      IF(MMAN.EQ.0)GO TO 2000
      IF(MAX.LE.IND)GO TO 2000
      MAN(4*IND-2)=XCOMP
      MAN(4*IND-1)=YCOMP
      MAN(4*IND)=ZCOMP
      INDEX=INDEX+1
      IF(INDEX.GT.IPTS)GO TO 3000
 2000 CONTINUE
C
C     UPDATE THE DACS
C
      LAUNCH(1)=XCOMP
      LAUNCH(2)=YCOMP
      LAUNCH(3)=ZCOMP
      LAUNCH(4)=RLB
      LAUNCH(5)=COSE
      LAUNCH(6)=SPO
      LAUNCH(7)=RI
      LAUNCH(8)=GAM
      LAUNCH(9)=EDOT
      LAUNCH(10)=THETAL
      LAUNCH(11)=RN
      WRITE(6,5001)(LAUNCH(II),I=1,11)                          SOFT-T
 5001 FORMAT(8H LAUNCH=,11F5.2)                                 SOFT-T
 1000 CALL SIMIDLE
      GO TO 1001                                                SOFT-T
      END
```

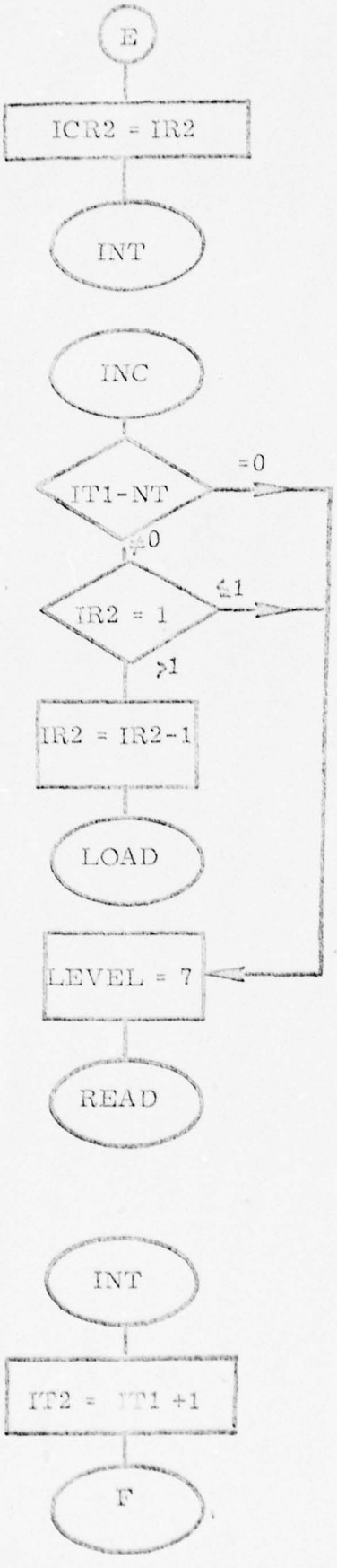

START
LEVEL = 7
IDA3 = 0
IT1 = 1
ICR2 = 29
IND = 0
INDEX = 0
KCK = -7
LAUNCH$_i$ = FIX (XDTGO$_i$)
i = 1 to 10
OUTPUT ALL ZEROES ON SENSE LINE B
INPUT SENSE LINE B INTO LINES
ACC = LINE E∧LINE S
ACC
=0
≠ 0
WRITE LAUNCH$_i$
i = 1 to 11
WAIT FOR COMPLETION
A

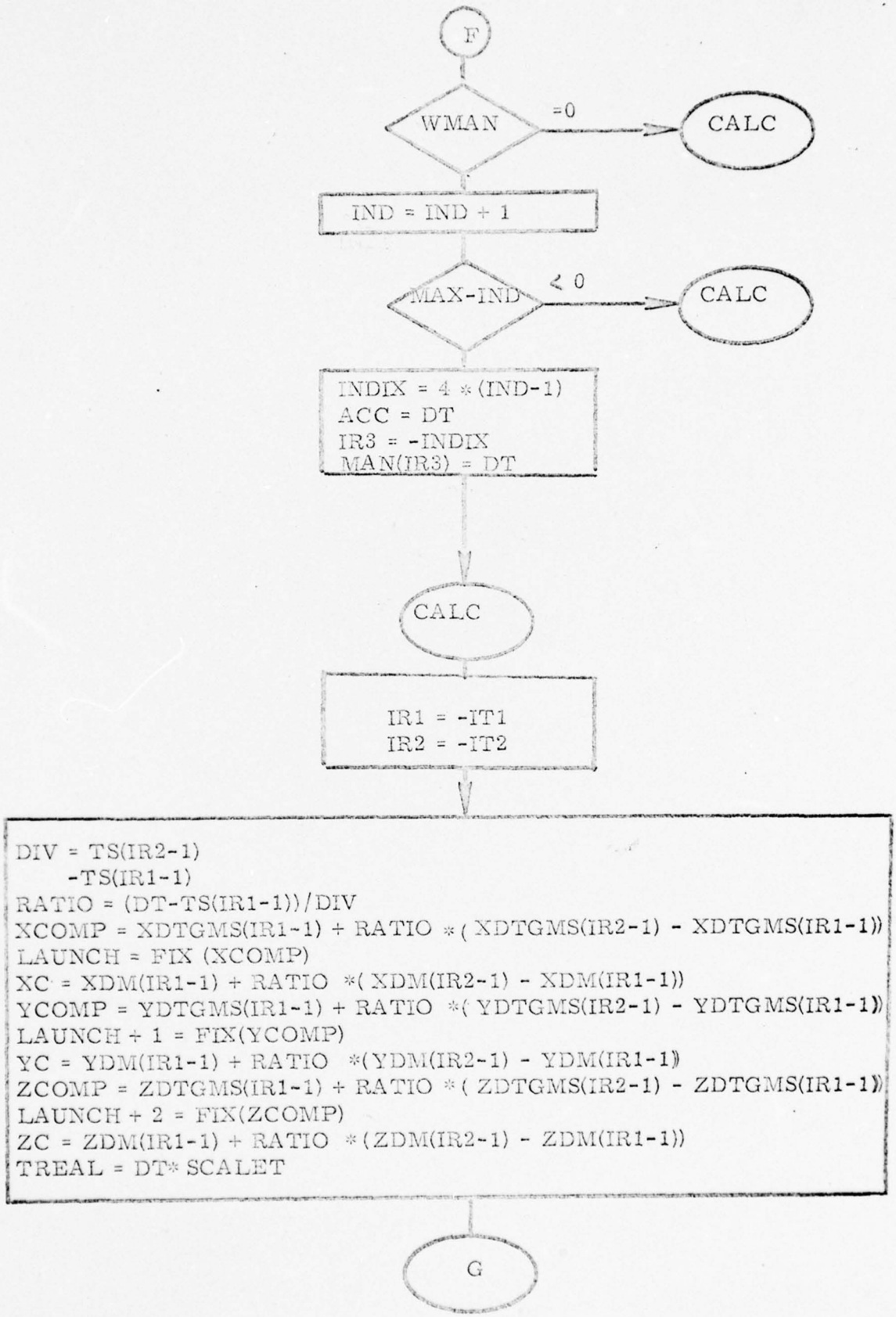

A
OUTPUT BIT 4
ON SENSE LINE B
INPUT SENSE LINE B
INTO LINE S
ACC = LINE D^ LINE S
=0
ACC
≠ 0
READ
OUTPUT BIT 8
ON SENSE LINE B
WMAN = 0
IR2 = 100
IR2 = IR2-1
>0
IR2
≤0
READ DX
i
i = 1 to 10
WAIT FOR
COMPLETION
B

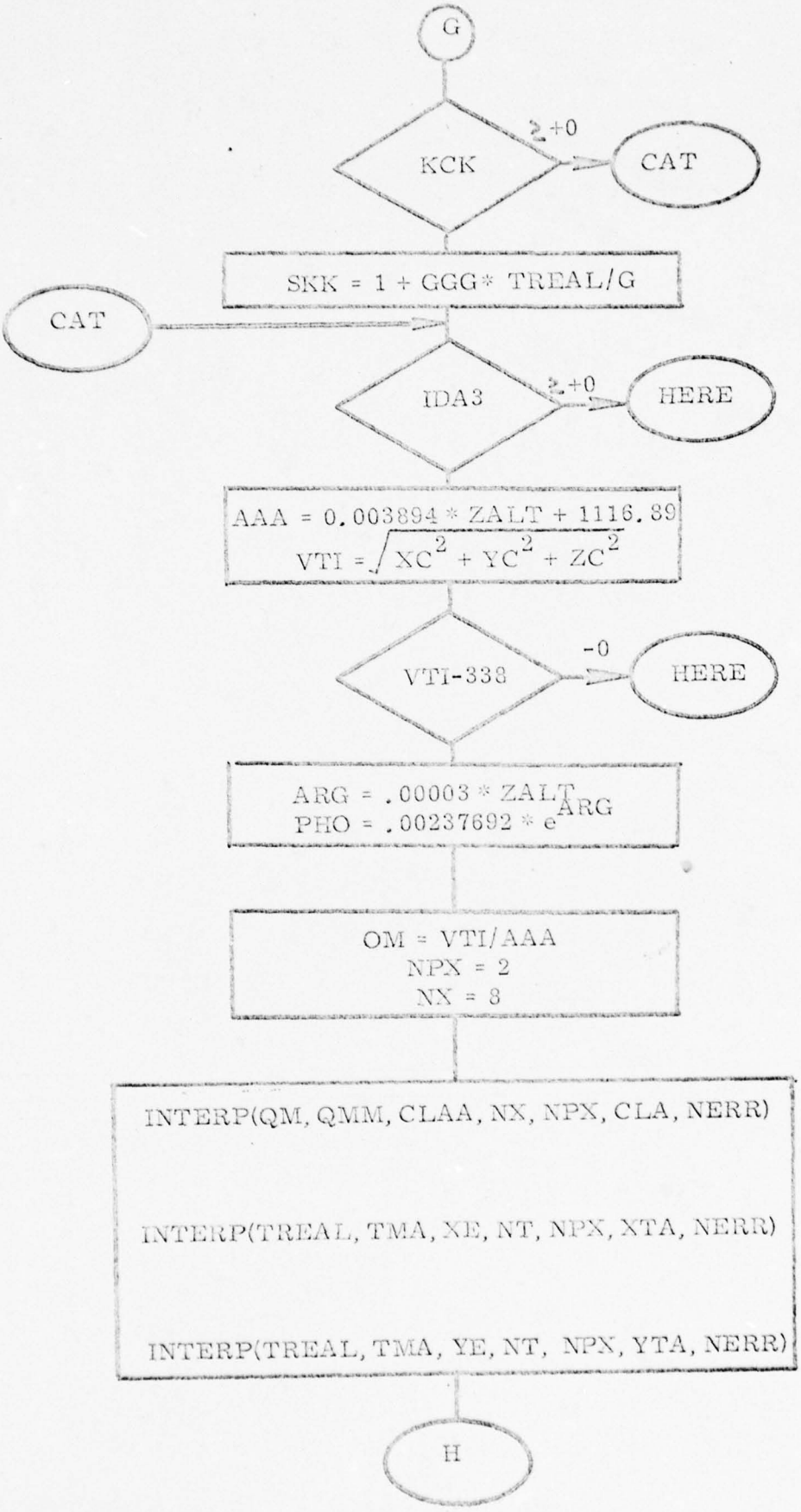

B
DX = FLOAT (DX)
DXG = DX
DXG = DXG * SCALEP
DY = FLOAT (DY)
DYG = DY
DYG = DYG * SCALEP
DZ = FLOAT (DZ)
DZG = DZ
DZG = DZG * SCALEP
DT = FLOAT (DT)
XDOT = FLOAT (XDOT)
XDO = XDOT
YDOT = FLOAT (YDOT)
YDO = YDOT
ZDOT = FLOAT (ZDOT)
ZDO = ZDOT
XXX = FLOAT (XXX)
YYY = FLOAT (YYY)
ZZZ = FLOAT (ZZZ)
IR2 = - INDEX
ACC = XXS (IR2)
ACC = ACC-DX
ACC
> 0
TPRIME
≤ 0
OUTPUT BIT 3
ON SENSE LINE B
C

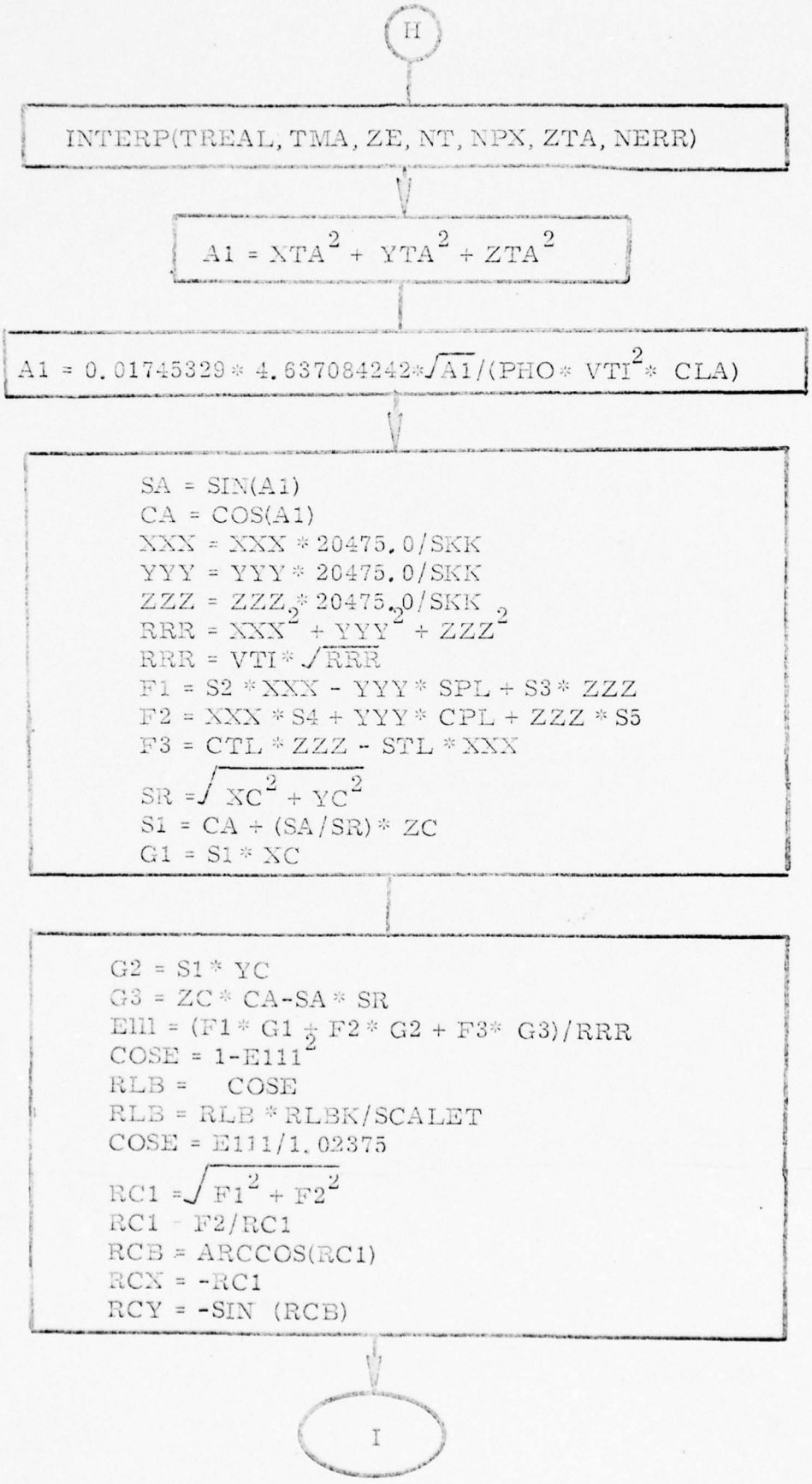

C
PPX(IR2) = DX
PPY(IR2) = DY
PPZ(IR2) = DZ
TIME(IR2) = OT
VMX(IR2) = XDOT
VMY(IR2) = YDOT
VMZ(IR2) = ZDOT
WMAN = 7
IDA3 = 1
TPRIME
ACC = DT/10 - G
ACC
>0
CKMISS
≤0
CKTIME
CKMISS
SKK = GGG + 1
KCK = 1
MISS = XDOT * DX
MISS = MISS + DY * YDOT
MISS = MISS + DZ * ZDOT
MISS
≤ 0
CKTIME
D

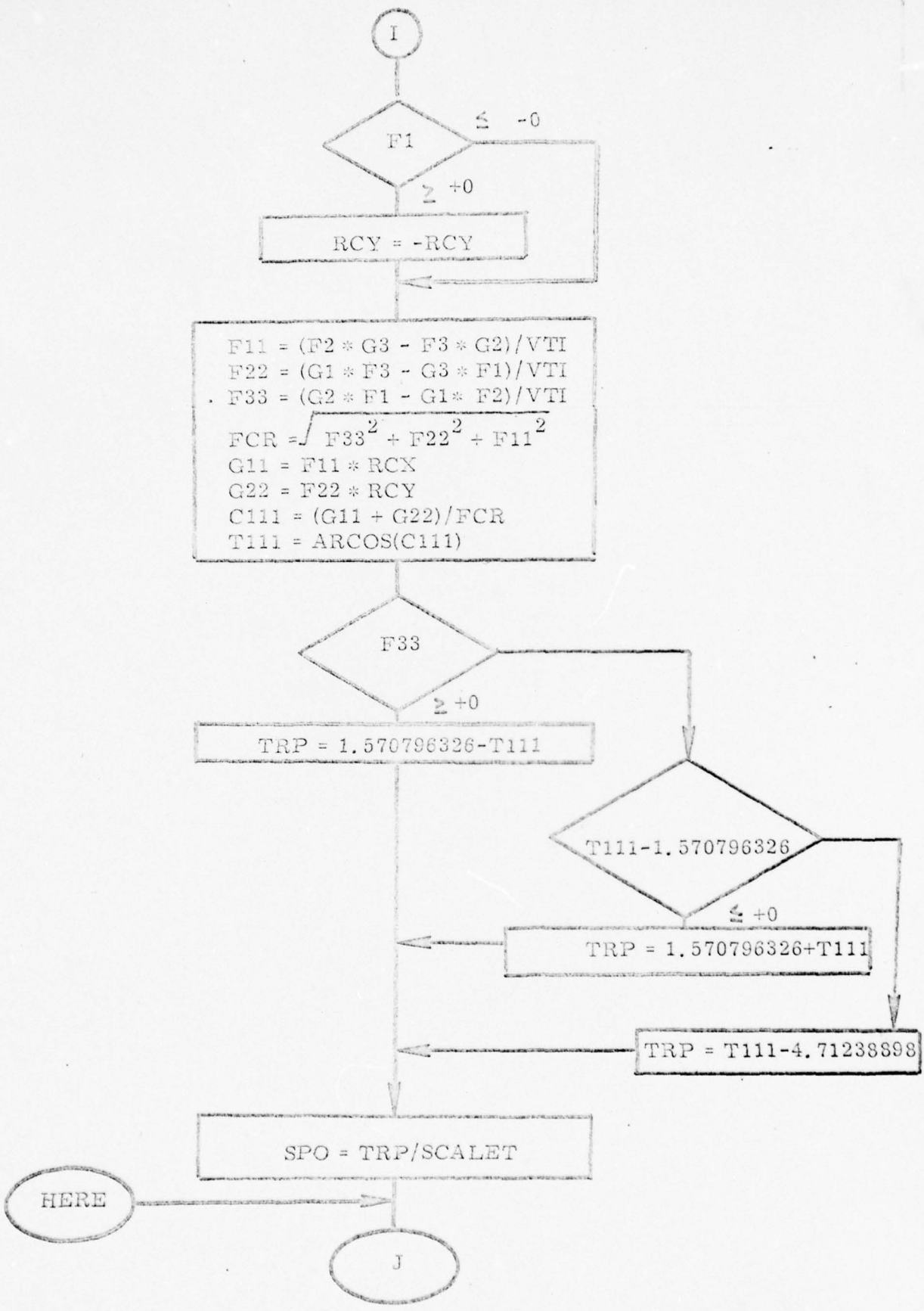

D
MISSED = DX
MISSED+1 = DY
MISSED+2 = DZ
MISSED+3 = XDOT
MISSED+4 = YDOT
MISSED+5 = ZDOT
MISSED+6 = MISS
OUTPUT BIT 7
ON SENSE LINE B
RETURN
CKTIME
LEVEL
>0
READ
≤0
IR2 - ICR2
DT-TS
<0
READ
LEVEL = 0
LOAD
E
DT-TS(30-IR2)
≥0
INC

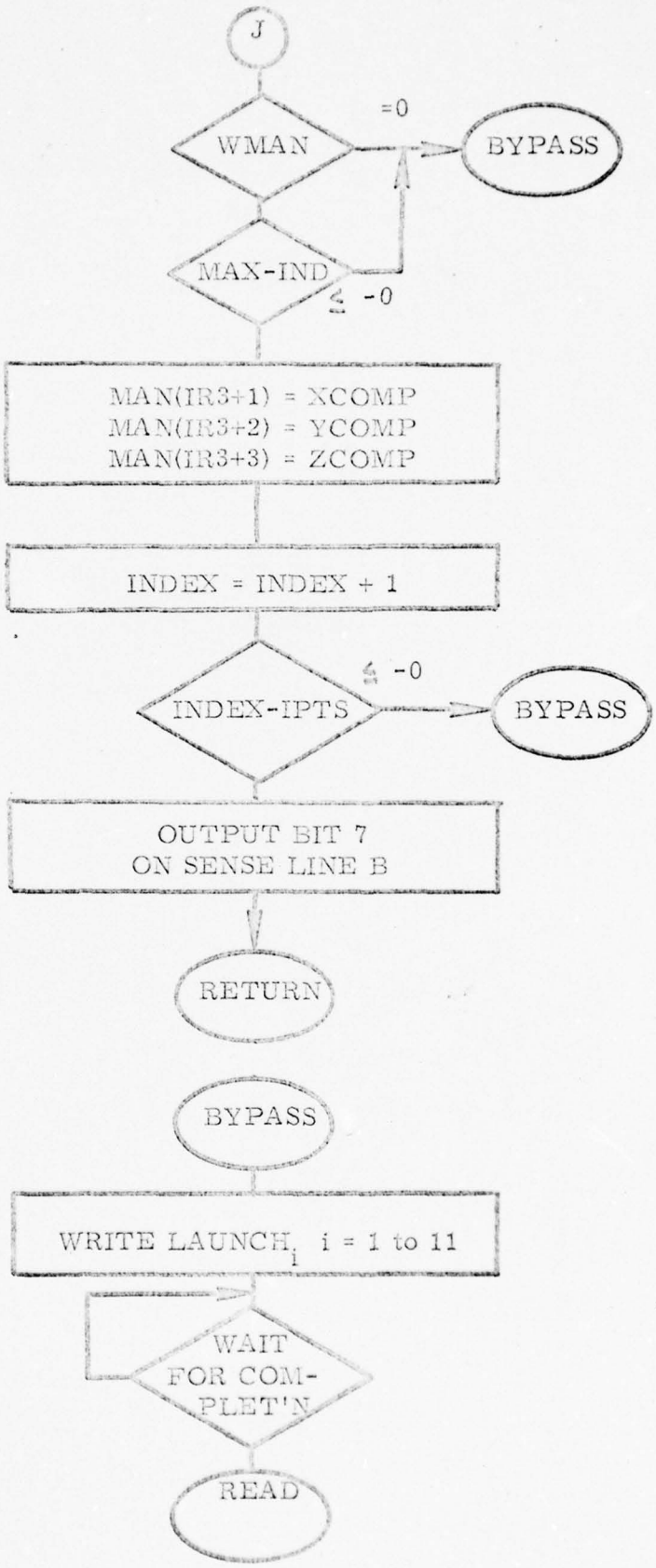

E
ICR2 = IR2
INT
INC
IT1-NT
=0
≠0
IR2 = 1
≤1
>1
IR2 = IR2-1
LOAD
LEVEL = 7
READ
INT
IT2 = IT1 +1
F


F
WMAN
=0
CALC
IND = IND + 1
MAX-IND
< 0
CALC
INDIX = 4 *(IND-1)
ACC = DT
IR3 = -INDIX
MAN(IR3) = DT
CALC
IR1 = -IT1
IR2 = -IT2
DIV = TS(IR2-1)
-TS(IR1-1)
RATIO = (DT-TS(IR1-1))/DIV
XCOMP = XDTGMS(IR1-1) + RATIO *(XDTGMS(IR2-1) - XDTGMS(IR1-1))
LAUNCH = FIX (XCOMP)
XC = XDM(IR1-1) + RATIO *(XDM(IR2-1) - XDM(IR1-1))
YCOMP = YDTGMS(IR1-1) + RATIO *(YDTGMS(IR2-1) - YDTGMS(IR1-1))
LAUNCH + 1 = FIX(YCOMP)
YC = YDM(IR1-1) + RATIO *(YDM(IR2-1) - YDM(IR1-1))
ZCOMP = ZDTGMS(IR1-1) + RATIO *(ZDTGMS(IR2-1) - ZDTGMS(IR1-1))
LAUNCH + 2 = FIX(ZCOMP)
ZC = ZDM(IR1-1) + RATIO *(ZDM(IR2-1) - ZDM(IR1-1))
TREAL = DT* SCALET
G


G
KCK
2.+0
CAT
SKK = 1 + GGG* TREAL/G
CAT
IDA3
2.+0
HERE
AAA = 0.003894 * ZALT + 1116.89
VTI = √(XC^2 + YC^2 + ZC^2)
VTI-338
-0
HERE
ARG = .00003 * ZALT
PHO = .00237692 * e^ARG
OM = VTI/AAA
NPX = 2
NX = 8
INTERP(QM, QMM, CLAA, NX, NPX, CLA, NERR)
INTERP(TREAL, TMA, XE, NT, NPX, XTA, NERR)
INTERP(TREAL, TMA, YE, NT, NPX, YTA, NERR)
H


H
INTERP(TREAL, TMA, ZE, NT, NPX, ZTA, NERR)
A1 = XTA^2 + YTA^2 + ZTA^2
A1 = 0.01745329 * 4.637084242*√A1/(PHO* VTI^2* CLA)
SA = SIN(A1)
CA = COS(A1)
XXX = XXX * 20475.0/SKK
YYY = YYY * 20475.0/SKK
ZZZ = ZZZ * 20475.0/SKK
RRR = XXX^2 + YYY^2 + ZZZ^2
RRR = VTI * √RRR
F1 = S2 *XXX - YYY* SPL + S3* ZZZ
F2 = XXX * S4 + YYY* CPL + ZZZ * S5
F3 = CTL * ZZZ - STL * XXX
SR = √XC^2 + YC^2
S1 = CA + (SA/SR)* ZC
G1 = S1 * XC
G2 = S1* YC
G3 = ZC* CA-SA* SR
E111 = (F1* G1 + F2* G2 + F3* G3)/RRR
COSE = 1-E111^2
RLB = COSE
RLB = RLB * RLBK/SCALET
COSE = E111/1.02375
RC1 = √F1^2 + F2^2
RC1 = F2/RC1
RCB = ARCCOS(RC1)
RCX = -RC1
RCY = -SIN (RCB)
I


I
F1
≤ -0
≥ +0
RCY = -RCY
F11 = (F2 * G3 - F3 * G2)/VTI
F22 = (G1 * F3 - G3 * F1)/VTI
F33 = (G2 * F1 - G1 * F2)/VTI
FCR = √(F33^2 + F22^2 + F11^2)
G11 = F11 * RCX
G22 = F22 * RCY
C111 = (G11 + G22)/FCR
T111 = ARCOS(C111)
F33
≥ +0
TRP = 1.570796326-T111
T111-1.570796326
≤ +0
TRP = 1.570796326+T111
TRP = T111-4.71238898
SPO = TRP/SCALET
HERE
J


J
WMAN
=0
BYPASS
MAX-IND
≤ -0
MAN(IR3+1) = XCOMP
MAN(IR3+2) = YCOMP
MAN(IR3+3) = ZCOMP
INDEX = INDEX + 1
INDEX-IPTS
≤ -0
BYPASS
OUTPUT BIT 7
ON SENSE LINE B
RETURN
BYPASS
WRITE LAUNCH$_i$ i = 1 to 11
WAIT
FOR COM-
PLET'N
READ

SUBROUTINE BITS

PURPOSE

The 16 bit discrete word sent by the AD-4 to the CDC-6600 is converted into bits by this program. The discrete word is periodically reconverted and updated in a COMMON block. On completion of a conversion the high order bit is stored in IBIT(1).

SUBROUTINES REQUIRED

- SUBROUTINE SIMIDLE

```
      SUBROUTINE BITS
      COMMON/ZIDIS2/IIN                                          SOFT-MOD
      COMMON/INTCOM/IBIT(60),MAXBIT
C
C     CONVERT IWORD TO BITS
C
      INTEGER OLDNUM
      DO 1 I=1,MAXBIT
    1 IBIT(I)=0
      OLDNUM=IIN
      I=1
    2 NEWNUM=OLDNUM/2
      IBIT(I)=OLDNUM-2*NEWNUM
      OLDNUM=NEWNUM
      I=I+1
      IF(OLDNUM.EQ.0.OR.I.LT.MAXBIT)GO TO 2
      CALL SIMIDLE
      RETURN                                                     SOFT-T
      END
```

SUBROUTINE ADFOUR AND ADC

PURPOSE

These programs together simulate AD-4 functions required by the STINGER real-time simulation. The programs functions as a table-look-up of ADC data required by SUBROUTINE REALT and SUBROUTINE FLIGHT. In addition, the programs transfers discrete data to the aforementioned subroutines.

SUBROUTINES REQUIRED

- SUBROUTINE BITS
- SUBROUTINE ADC

```
      SUBROUTINE ADFOUR                                                 SOFT-T
      REAL MAN(200),MISSED(7),MISS,LAUNCH                               SOFT-T
      INTEGER WMAN                                                      SOFT-T
      DIMENSION FIN(10),TS(30)                                          SOFT-T
      EQUIVALENCE (TS(1),TMAS(1)),(MAN(1),XMAN(1)),(MISSED(1),XMISS(1))
      EQUIVALENCE (FIN(1),DX),(FIN(2),DY),(FIN(3),DZ),(FIN(4),DT)       SOFT-T
      EQUIVALENCE (FIN(5),XDOT),(FIN(6),YDOT),(FIN(7),ZDOT)             SOFT-T
      EQUIVALENCE (FIN(8),XXX),(FIN(9),YYY),(FIN(10),ZZZ)               SOFT-T
      COMMON/COMA/LEVEL,IPTS,XXS(50),XDTGO,YDTGO,ZDTGO,RLB,COSE,SPO,RI, SOFT-T
     *BAM,EDOT,THETAL,RN,                                               SOFT-T
     *          PPX(50),PPY(50),PPZ(50),TIME(50),TMAS(30),XDTGMS(30)    SOFT-T
     *         ,YDTGMS(30),ZDTGMS(30),XMAN(4,50),XMISS(7),NT            SOFT-T
     *         ,XCOMP,YCOMP,ZCOMP,TAMA(30),DELTAR(30),VM(30),G,GGG       SOFT-T
     *,XOO,YOO,ZOO,DXG,DYG,DZG,S2,S3,S4,S5,XDM(30),YDM(30),ZDM(30),      SOFT-T
     *RLBK,SCALEP,F1,F2,F3,G1,G2,G3,XC,YC,ZC,S1,RRR,SR,SPL,CTL,STL,      SOFT-T
     *CPL,A1,VTI,XE(30),YE(30),ZE(30),ZALT,NERR,CLA,NPX,NX,CLAA(10),     SOFT-T
     *PHC,ARG,AAA,SCALET,TREAL,TMA(30),XTA,YTA,ZTA,SCALEV,QMM(10),QM     SOFT-T
     *,SA,CA,VMX(50),VMY(50),VMZ(50)                                    SOFT-T
      COMMON/ZADC1/ADIN(16)                                             SOFT-MOD
      COMMON/ZDAC1/LAUNCH(11)                                           SOFT-MOD
      COMMON/ZODIC2/IOUT                                                SOFT-MOD
      COMMON/ZIDIS2/IIN                                                 SOFT-MOD
      DATA ITIME/-100/                                                  SOFT-T
      ITIME=ITIME+1                                                     SOFT-T
C     STATIC CHECK OK                                                   SOFT-T
      IF(ITIME.EQ.-50)IIN=32                                            SOFT-T
C     SEND ICS                                                          SOFT-T
      IF(ITIME.EQ.-50)CALL ADC(ITIME)                                   SOFT-T
C     RAMP UP SIGNAL                                                    SOFT-T
      IF(ITIME.EQ.-40)IIN=64                                            SOFT-T
C                                                                       SOFT-T
      CALL BITS                                                         SOFT-T
C     UPDATE ADC                                                        SOFT-T
      CALL ADC(ITIME)                                                   SOFT-T
      RETURN                                                            SOFT-T
      END                                                               SOFT-T
```

```
      SUBROUTINE ADC(ITIME)                                              SOFT-T
      REAL MAN(200),MISSED(7),MISS,LAUNCH                                SOFT-T
      INTEGER NMAN                                                       SOFT-T
      DIMENSION FIN(10),TS(30)                                           SOFT-T
      EQUIVALENCE (TS(1), TMAS(1)),(MAN(1),XMAN(1)),(MISSED(1),XMISS(1))
      EQUIVALENCE (FIN(1),DX),(FIN(2),DY),(FIN(3),DZ),(FIN(4),DT)        SOFT-T
      EQUIVALENCE (FIN(5),XDOT),(FIN(6),YDOT),(FIN(7),ZDOT)              SOFT-T
      EQUIVALENCE (FIN(8),XXX),(FIN(9),YYY),(FIN(10),ZZZ)                SOFT-T
      COMMON/COMA/LEVEL,IPTS,XXS(50),XDTGO,YDTGO,ZDTGO,RLB,COSE,SPO,RI,  SOFT-T
     *GAM,EDOT,THETAL,RN,                                                SOFT-T
     *          PPX(50),PPY(50),PPZ(50),TIME(50),TMAS(30),XDTGMS(30)     SOFT-T
     *         ,YDTGMS(30),ZDTGMS(30),XMAN(4,50),XMISS(7),NT             SOFT-T
     *         ,XCOMP,YCOMP,ZCOMP,TAMA(30),DELTAR(30),VM(30),G,GGG        SOFT-T
     *,XDO,YDO,ZDO,DXG,DYG,DZG,S2,S3,S4,S5,XDM(30),YDM(30),ZDM(30),       SOFT-T
     *RLBK,SCALEP,F1,F2,F3,G1,G2,G3,XC,YC,ZC,S1,RRR,SR,SPL,CTL,STL,       SOFT-T
     *CPL,A1,VTI,XE(30),YE(30),ZE(30),ZALT,NERR,CLA,NPX,NX,CLAA(10),      SOFT-T
     *PHO,ARG,AAA,SCALET,TREAL,TMA(30),XTA,YTA,ZTA,SCALEV,QMM(10),QM      SOFT-T
     *,SA,CA,VMX(50),VMY(50),VMZ(50)                                     SOFT-T
      COMMON/ZADC1/ADIN(10)                                              SOFT-MOD
      COMMON/ZDAC1/LAUNCH(11)                                            SOFT-MOD
      COMMON/ZODIS2/IOUT                                                 SOFT-MOD
      COMMON/ZIDIS2/IIN                                                  SOFT-MOD
C                                                                        SOFT-T
C     PROGRAM TO UPDATE ADC INPUTS                                       SOFT-T
C                                                                        SOFT-T
      DIMENSION A(10,500)                                                SOFT-T
      IF(ITIME.EQ.-50)KTIME=1                                            SOFT-T
      IF(ITIME.GE.0)KTIME=KTIME+1                                        SOFT-T
      IF(KTIME.GT.500)WRITE(6,1000)
C                                                                        SOFT-T
C                                                                        SOFT-T
C                                                                        SOFT-T
C     DEFINE THE A ARRAY HERE                                            SOFT-T
      IF (KTIME.GE.2) GO TO 101
      DO 50 I=1,10                                                       SOFT-T
      DO 50 KDUM=1,500                                                   SOFT-T
      A(KDUM,I)=0.0                                                      SOFT-T
   50 CONTINUE                                                           SOFT-T
  101 CONTINUE
C                                                                        SOFT-T
      DO 100 I=1,10                                                      SOFT-T
      ADIN(I)=A(KTIME,I)                                                 SOFT-T
  100 CONTINUE                                                           SOFT-T
      RETURN                                                             SOFT-T
 1000 FORMAT(15H ERROR IN KTIME)                                         SOFT-T
      END                                                                SOFT-T
```

```
      SUBROUTINE SIMSTOP                                                SOFT-T
      RETURN                                                            SOFT-T
      END                                                               SOFT-T


      SUBROUTINE SHOLD                                                  SOFT-T
      RETURN                                                            SOFT-T
      END                                                               SOFT-T


      SUBROUTINE SIMRUN(ISTAT)                                          SOFT-T
      ISTAT=0                                                           SOFT-T
      RETURN                                                            SOFT-T
      END                                                               SOFT-T


      SUBROUTINE SIMIDLE                                                SOFT-T
      RETURN                                                            SOFT-T
      END                                                               SOFT-T
```

APPENDIX E

DADIOS CHECKOUT PROGRAMS

BOK
DYNAMICS, INC.

DADIOS CHECKOUT PROGRAMS

This appendix contains special programs for pre-real-time checkout. These programs provide a quick method of testing discretes, DACs and ADCs. Usage of each program is described in the computer code.

```
      PROGRAM TRDISI(OUTPUT,HFILE,TAPE6=OUTPUT)
C
C     PROGRAM TO INDIVIDUALLY TEST DADIOS DISCRETES FROM AD/4 TO
C     CDC/6600. THIS IS ACCOMPLISHED BY PATCHING LOGIC 1 TO THE DESIRED
C     AD/4 TRUNK LINES. EACH TIME THE  AD/4 PATCHING IS CHANGED THE
C     CDC/6600 RECORDS THE BIT PATTERN FOR COMPARISON. A RECORD OF THE
C     BIT PATTERNS IS AVAILABLE THROUGH OPERATOR AID OR THE LINE PRINTER
C
C     PROGRAM VARIABLES
C        IERR           ERROR CODE FOR RESERVATION
C                       0=NOERROR, GT.0=RESERVATION ERROR
C        ISTAT          REAL TIME MODE
C                       0=IN REAL TIME, ISTAT.GT.0 NOT IN REAL TIME
C        IDUM1          DISCRETE WORD TRANSMITTED FROM AD/4 TO CDC/6600
C        ICNT1          TIME SINCE LAST BIT WAS CHANGED (SECONDS)
C
C     DADIOS PATCHING REQUIREMENTS (ONE OF THE FOLLOWING)
C
C          TRUNKING              FORTRAN                 AD/4 LOGIC
C        W-50 TO W-50      FOR /IDIS2/1,IIDIS      TR00-TR07 AND TR20-TR27
C        W-50 TO W-51      FOR /IDIS2/2,IIDIS      TR00-TR07 AND TR20-TR27
C        W-52 TO W51       FOR /IDIS2/2,IIDIS      TR40-TR47 AND TR60-TR67
C        W-52 TO W50       FOR /IDIS2/1,IIDIS      TR40-TR47 AND TR60-TR67
C
      COMMON/INTCOM/ICNT1,IDUM1,ITEMP,IBIT(60)
      INTERRUPT(I=1,R=10,T=100000)
      COMMON/*IDIS2/2,IIDIS
C
C     INITIALIZATION
C
      ITEMP=0
      CALL RESERVE(IERR)
      ICNT1=0
      WRITE(6,1000)IERR
      IF(IERR.NE.0)STOP
C
C     REAL TIME
C
      CALL SIMRUN(ISTAT)
      WRITE(6,2000)ISTAT
      WRITE(6,6000)
      WRITE(6,5000)
      IF(ISTAT.GT.0)STOP
      CALL REMARK(17H JOB IN REAL TIME)
   25 CONTINUE
      CALL BHOLD
      CALL REMARK(15H RETURN TO MAIN)
      CALL CDCDIS( 54 BITS,IDUM1)
      CALL BITS
      WRITE(6,4000)IDUM1,IDUM1,(IBIT(KK),KK=1,16),ICNT1
      ICNT1=0
      CALL SIMGO
      GO TO 25
 1000 FORMAT(2H1RESERVATION ERROR CODE=,O20)
 2000 FORMAT(18H REAL TIME STATUS=,O20)
 4000 FORMAT(5X,O10,I10,5X,16I1,I10,*  TR=.....*)
 5000 FORMAT(*0         OCTAL   BASE TEN      ......BITS....       TIME*)
```

```
 6000 FORMAT(*)                    RECORD OF DATA RECIVED *)
      STOP
      END


      SUBROUTINE BITS
C
C     PROGRAM TO CONVERT  DISCRETE WORD TO BITS
C
      COMMON/INTCOM/ICNT1,IDUM1,IDUMY,IBIT(60)
      INTEGER OLDNUM
      MAXBIT=15
      DO 1 I=1,MAXBIT
    1 IBIT(I)=0
      OLDNUM=IDUM1
      DO 2 I=1,MAXBIT
      NEWNUM=OLDNUM/2
      IBIT(I)=OLDNUM-2*NEWNUM
      OLDNUM=NEWNUM
    2 CONTINUE
      IHALF=MAXBIT/2
      DO 3 I=1,IHALF
      ITEMP=IBIT(I)
      IBIT(I)=IBIT(MAXBIT+1-I)
    3 IBIT(MAXBIT+1-I)=ITEMP
      RETURN
      END


      SUBROUTINE SUB1
C
C     REAL TIME INTERRUPT SUBROUTINE
C
      COMMON/INTCOM/ICNT1,IDUM1,ITEMP
      COMMON/*IDIS2/2,IIDIS
      ICNT1=ICNT1+1
      IDUM1=IIDIS
      IF(ITEMP.EQ.IDUM1)GO TO 10
      ITEMP=IIDIS
      CALL SIMHOLD
   10 CONTINUE
      CALL SIMIDLE
      END


      RTREE TRDISI(0),SUB1(1)
      GLOBAL INTCOM
      END
```

```
      PROGRAM TRDISO(OUTPUT,HFILE,TAPE5=OUTPUT)
C
C     PROGRAM TO INDIVIDUALLY TEST DADIOS DISCRETES FROM CDC/6600 TO AD/4
C     THIS IS ACCOMPLISHED BY LETTING THE CDC/6600 SEND A BIT AND PAUSE.
C     THE BIT CAN THEN BE VERIFIED AT THE AD/4 CONSOLE BY APPROPRIATE
C     PATCHING TO AN INDICTOR LIGHT ON THE DIGITAL LOGIC BOARD. THE NEXT,
C     AND EACH SUCCEDING, BIT IS RAISED BY A GO COMMAND GIVEN THROUGH DDS.
C
C     PROGRAM VARIABLES
C        IERR          ERROR CODE FOR RESERVATION
C                      0=NOERROR, GT.0=RESERVATION ERROR
C        ISTAT         REAL TIME MODE
C                      0=IN REAL TIME, ISTAT.GT.0 NOT IN REAL TIME
C        IDUM1         DISCRETE WORD TRANSMITTED FROM CDC/6600 TO AD/4
C        ICNT1         TIME SINCE LAST BIT WAS CHANGED (SECONDS)
C
C     DADIOS PATCHING REQUIREMENTS (ONE OF THE FOLLOWING)
C
C          TRUNKING              FORTRAN                AD/4 LOGIC
C        V-51 TO W-60      FOR /ODIS2/1,IODIS      TR10-TR17 AND TR30-TR37
C        V-53 TO W-60      FOR /ODIS2/1,IODIS      TR50-TR57 AND TR70-TR77
C        V-51 TO W-51      FOR /ODIS2/2,IODIS      TR10-TR17 AND TR30-TR37
C        V-53 TO W-61      FOR /ODIS2/2,IODIS      TR50-TR57 AND TR70-TR77
C
      COMMON/INTCOM/ICNT1,IDUM1,IBIT(60)
      INTERRUPT(I=1,R=20,T=100000)
      COMMON/*ODIS2/2,IODIS
C
C     INITIALIZATION
C
      ICNT1=0
      ICOUNT=0
      CALL RESERVE(IERR)
      WRITE(6,10000)IERR
      IF(IERR.NE.0)STOP
C
C     REAL TIME
C
      CALL SIMRUN(ISTAT)
      CALL REMARK(17H JOB IN REAL TIME)
      WRITE(6,20000)ISTAT
      IF(ISTAT.GT.0)STOP
      WRITE(6,60000)
      WRITE(6,50000)
   25 CONTINUE
      CALL BHOLD
      IDUM1=2**ICOUNT
      CALL BITS
      WRITE(6,40000)IDUM1,IDUM1,(IBIT(KK),KK=1,16),ICNT1
      ICOUNT=ICOUNT+1
      IF(ICOUNT.EQ.17)ICOUNT=0
C     PAUSE
C     CALL OCTDIS( 5H BITS,IDUM1)
C     CALL REMARK(15H RETURN TO MAIN)
      ICNT1=0
      CALL SIMGO
      GO TO 25
```

```
1000 FORMAT(24H RESERVATION ERROR CODE=,O20)
2000 FORMAT(18H REAL TIME STATUS=,O20)
4000 FORMAT(5X,O20,I10,5X,16I1,I10)
5000 FORMAT(*0          OCTAL    BASE TEN       ......BITS....        TIME*)
6000 FORMAT(*1                   RECORD OF DATA SENT *)
     STOP
     END


     SUBROUTINE SUB1
C
C    REAL TIME INTERRUPT SUBROUTINE
C
     COMMON/INTCOM/ICNT1,IDUM1,IBIT(5)
     COMMON/*ODIS2/2,IODIS
     ICNT1=ICNT1+1
     IODIS=IDUM1
     IF(ICNT1.GT.5)CALL SIMHOLD
     CALL SIMIOLI
     END


     SUBROUTINE BITS
C
C    PROGRAM TO CONVERT DISCRETE WORD TO BITS
     COMMON/INTCOM/ICNT1,IDUM1,IBIT(5)
     INTEGER OLDNUM
     MAXBIT=16
     DO 1 I=1,MAXBIT
   1 IBIT(I)=0
     OLDNUM=IDUM1
     DO 2 I=1,MAXBIT
     NEWNUM=OLDNUM/2
     IBIT(I)=OLDNUM-2*NEWNUM
     OLDNUM=NEWNUM
   2 CONTINUE
     IHALF=MAXBIT/2
     DO 3 I=1,IHALF
     ITEMP=IBIT(I)
     IBIT(I)=IBIT(MAXBIT+1-I)
   3 IBIT(MAXBIT+1-I)=ITEMP
     RETURN
     END


     RTREE TRDISO(0),SUB1(1)
     GLOBAL INTCOM
     END
```

```
      PROGRAM TRDISIO(OUTPUT,MFILE,TAPE6=OUTPUT)
C
C     PROGRAM TO TEST DISCRETE WORDS BETWEEN AD/4 AND CDC/6600. THIS
C     TASK IS ACCOMPLISHED BY TURNING AROUND BITS SENT BY THE CDC/6600
C     AND COMPARING THEM UPON RETURN. THE PROGRAM TEST ALL POSSIBLE BIT
C     PATTERNS FOR A 16 BIT LINE.
C
C     THE HIGH ORDER CDC-6600 BIT CORRESPONDS TO TR0X, WHERE X=0,2,4,6
C
C     PROGRAM VARIABLES
C        IERR          ERROR CODE FOR RESERVATION
C                      0=NOERROR, GT.0=RESERVATION ERROR
C        ISTAT         REAL TIME MODE
C                      0=IN REAL TIME, ISTAT.GT.0 NOT IN REAL TIME
C        IIDIS         CDC-6600 SENSE LINE DISCRETE(16 BIT)  IIDIS=IBACK
C        IODIS         CDC-6600 CONTROL LINE DISCRETE(16 BIT) IODIS=IOUT
C        MAX           DEC. EQUIVALENT OF 16 BITS ALL EQUAL ONE
C        LOOP          NUMBER OF INTERRUPTS BEFORE EQUALITY OF BITS
C        LINE          NUMBER OF LINES OF PRINTOUT IN EXECUTION
C
C     DADIOS PATCHING REQUIREMENTS (ONE IIDIS AND ONE IODIS)
C
C          TRUNKING              FORTRAN              AD/4 LOGIC
C        V-50 TO W-50      FOR /IDIS2/1,IIDIS     TR00-TR07 AND TR20-TR27
C        V-52 TO W-50      FOR /IDIS2/1,IIDIS     TR40-TR47 AND TR60-TR67
C        V-50 TO W-51      FOR /IDIS2/2,IID S     TR00-TR07 AND TR20-TR27
C        V-52 TO W-51      FOR /IDIS2/2,IIDIS     TR40-TR47 AND TR60-TR67
C
C        V-51 TO W-60      FOR /ODIS2/1,IODIS     TR10-TR17 AND TR30-TR37
C        V-53 TO W-60      FOR /ODIS2/1,IODIS     TR50-TR57 AND TR70-TR77
C        V-51 TO W-51      FOR /ODIS2/2,IODIS     TR10-TR17 AND TR30-TR37
C        V-53 TO W-51      FOR /ODIS2/2,IODIS     TR50-TR57 AND TR70-TR77
C
C
      COMMON/INTCOM/IOUT,LOOP,MAX,IBACK
      INTERRUPT(I=1,R=10,T=500)
      COMMON/*IDIS2/2,IIDIS
      COMMON/*ODIS2/2,IODIS
      CALL RESERVE(IERR)
      WRITE(6,1000)IERR
      IF(IERR.NE.0)STOP
C
C     INITIALIZATION
C
      MAX=2**16-1
      IOUT=0
      LOOP=0
      LINES=0
C
C     REAL TIME
C
      CALL SIMRUN(ISTAT)
      WRITE(6,2000)ISTAT
      IF(ISTAT.GT.0)STOP
      CALL REMARK(17H JOB IN REAL TIME)
      WRITE(6,6000)
      WRITE(6,5000)
```

```
   25 CONTINUE
      CALL EHOLD
      WRITE(6,4000)IOUT,IOUT,IBACK,IBACK,LOOP
      CALL SIMGO
      GO TO 25
      LINES = LINES + 1
      IF(LINES.GT.200)STOP
      WRITE(6,3000)
      CALL RLMARK(15H RETURN TO MAIN)
 1000 FORMAT(24H1RESERVATION ERROR CODE=,O20)
 2000 FORMAT(18H REAL TIME STATUS=,O20)
 3000 FORMAT(1H0,*PROGRAM TERMINATED NORMALLY*)
 4000 FORMAT(10X,O11,I15,3X,O10,I15,I15)
 5000 FORMAT(6X,* IOUT(OCTAL)    IOUT(DECIMAL)    IBACK(OCTAL)  IBACK(DECI
     1MAL)  LOOP(DECIMAL)*/)
 6000 FORMAT(///,35X,*ERRORS DETECTED*//)
      STOP
      END


      SUBROUTINE SUB1

C
C     REAL TIME INTERRUPT SUBROUTINE
C
      COMMON/INTCOM/IOUT,LOOP,MAX,IBACK
      COMMON/*IDIS2/2,IIDIS
      COMMON/*ODIS2/2,IODIS
      IODIS=IOUT
      IBACK=IIDIS
      IF(IOUT.NE.IBACK)GO TO 10
      IF(IOUT.EQ.MAX)IOUT=0
      IOUT=IOUT+1
      LOOP=0
   10 LOOP=LOOP+1
      IF(LOOP.EQ.10)CALL SIMHOLD
      CALL SIMIDLE
      END



      RTREE TRDISIO(0),SUB1(1)
      GLOBAL INTCOM
      END
```

```
      PROGRAM TRALGI(OUTPUT,HFILE,TAPE6=OUTPUT)
C
C     PROGRAM TO TEST DADIOS ADDS FROM AD/4 TO CDC/6600. THIS IS
C     ACCOMPLISHED BY PATCHING AN ANALOG SIGNAL TO THE DESIRED AD/4
C     TRUNK LINE. EACH TIME THE AD/4 SIGNAL CHANGES THE CDC/6600 RECORDS
C     THE NEW ANALOG SIGNAL.
C
C
C     PROGRAM VARIABLES
C        IERR          ERROR CODE FOR RESERVATION
C                      0=NOERROR, GT.0=RESERVATION ERROR
C        ISTAT         REAL TIME MODE
C                      0=IN REAL TIME, ISTAT.GT.0 NOT IN REAL TIME
C        PERCENT       PERCENT CHANGE REQUIRED IN ADC VALUE BEFORE NEW
C                      ADC VALUE IS RECORDED BY CDC/6600
C        LINE          NUMBER OF LINES OF PRINTOUT IN EXECUTION
C
C
C     NOTE ADDS  ARE IN GROUPS OF 16, 1-16, 17-32, 33-48, 49-64.
C     FLOATING POINT ANALOG SIGNALS ARE SCALED GE -1.0 AND LE +1.0.
C     INTEGER ANALOG SIGNALS ARE SCALED GE -32767 AND LE +32767(14 BIT).
C
C     DADIOS PATCHING REQUIREMENTS (ONE OF THE FOLLOWING)
C
C          TRUNKING              FORTRAN                AD/4 LOGIC
C        W-03 TO W-06     FOR /*ADC1/49,ADC      TR10-TR17 AND TR30-TR37
C
      COMMON/INTCOM/BACK,LOOP,PERCENT,TEMP
      INTERRUPT(I=1,R=10,T=5000)
      COMMON/*ADC1/49,ADC
C
C     INITIALIZATION
C
      PERCENT=.05
      LINE=0
      TEMP=0.0
      CALL RESERVE(IERR)
      WRITE(6,1000)IERR
      IF(IERR.NE.0)STOP
C
C     REAL TIME
C
      CALL SIMRUN(ISTAT)
      WRITE(6,2000)ISTAT
      IF(ISTAT.GT.0)STOP
      CALL REMARK(17H JOB IN REAL TIME)
      WRITE(6,500)
   25 CONTINUE
      CALL SHOLD
      LINE=LINE+1
      IF(LINE.GT.200)STOP
      LOOP=0
      WRITE(6,3000)BACK,LOOP
      CALL SIMGO
      GO TO 25
 1000 FORMAT(24H1RESERVATION ERROR CODE=,O20)
 2000 FORMAT(18H REAL TIME STATUS=,O20)
```

```
 3000 FORMAT(5X,F10.4,I10)
 6000 FORMAT(*1     RECORD OF DATA RECIVED BY CDC/6600*//)
60    STOP
      END


      SUBROUTINE SUB1
C
C     REAL TIME INTERRUPT SUBROUTINE
C
      COMMON/INTCOM/BACK,LOOP,PERCENT,TEMP
      COMMON/*ADC1/+9,ADC
      LOOP=LOOP+1
      BACK=ADC
      PCHANGE=ABS(ABS(TEMP)-ABS(BACK))/ABS(TEMP)
      IF(PCHANGE.GT.PERCENT)GO TO 10
      TEMP=ADC
      CALL SIMHOLD
   10 CONTINUE
      CALL SIMIDLE
      END


      RTREE TRALGI(0),SUB1(1)
      GLOBAL INTCOM
      END
```

```
      PROGRAM TRALGO(OUTPUT,HFILE,TAPE6=OUTPUT)
C
C     PROGRAM TO INDIVIDUALLY TEST DADIOS DACS FROM CDC/6600 TO AD/4.
C     THIS IS ACCOMPLISHED BY LETTING THE CDC/6600 GENERATE A FUNCTION
C     F=F(TIME). THE FUNCTION CAN BE VERIFIED AT THE AD/4 CONSOLE BY
C     APPROPRIATE PATCHING TO A RECORDER.
C
C
C     PROGRAM VARIABLES
C        IERR          ERROR CODE FOR RESERVATION
C                      (=NOERROR, GT.0=RESERVATION ERROR
C        ISTAT         REAL TIME MODE
C                      0=IN REAL TIME, ISTAT.GT.0 NOT IN REAL TIME
C        TIME          INDEPENDENT VARIABLE WHICH IS PROPORTIONAL TO
C                      REAL TIME
C        DAC           THE DAC VARIABLE, NOTE OUT=DAC
C
C
C     NOTE DACS ARE IN GROUPS OF 16, 1-16, 17-32, 33-48, 49-64.
C     FLOATING POINT ANALOG SIGNALS ARE SCALED GE -1.0 AND LE +1.0.
C     INTEGER ANALOG SIGNALS ARE SCALED GE -32767 AND LE +32767(14 BIT).
C
C     DADIOS PATCHING REQUIREMENTS (ONE OF THE FOLLOWING)
C
C          TRUNKING              FORTRAN                AD/4 LOGIC
C        W-13 TO V-07      FOR /*DAC1/49,DAC      TR50-TR57 AND TR70-TR77
C
      COMMON/INTCOM/OUT,TIME,LOOP
      INTERRUPT(I=1,R=10,T=5000)
      COMMON/*DAC1/49,DAC
C
C     INITIALIZATION
C
      TIME=0.0
      CALL RESERVE(IERR)
      WRITE(6,1000)IERR
      IF(IERR.NE.0)STOP
C
C     REAL TIME
C
      CALL SIMRUN(ISTAT)
      WRITE(6,2000)ISTAT
      IF(ISTAT.GT.0)STOP
      CALL REMARK(17H JOB IN REAL TIME)
   25 CONTINUE
      CALL BHOLD
 1000 FORMAT(24H1RESERVATION ERROR CODE=,O20)
 2000 FORMAT(18H REAL TIME STATUS=,O20)
 3000 FORMAT(5X,F10.4,I10)
      STOP
      END
```

```
      SUBROUTINE SUB1
C
C     REAL TIME INTERRUPT SUBROUTINE
C
      COMMON/INTCOM/OUT,TIME,LOOP
      COMMON/*DAC1/49,DAC
      TIME=TIME+.1
      IF(TIME.GT.6.28)TIME=0.0
      OUT=SIN(TIME)
      DAC=OUT
      CALL SIMIDLE
      END


      ATREE TRAL,SUB(0),SUB1(1)
      GLOBAL INTCOM
      END
```

```
      PROGRAM TRALGIO(OUTPUT,HFILE,TAPE6=OUTPUT)
C
C     PROGRAM TO TEST ANALOG SIGNALS BETWEEN AD/4 AND CDC/6600. THIS TASK
C     IS ACCOMPLISHED BY TURNING THE ANALOG SIGNAL AROUND AT THE AD/4 AND
C     COMPARING DIFFERENCE UPON RETURN TO THE CDC/6600. THE PROGRAM TEST
C     FOR ERRORS GREATER THAN FIVE PERCENT.
C
C     NOTE ADCS AND DACS ARE IN GROUPS OF 16, 1-16, 17-32, 33-48, 49-64.
C     FLOATING POINT ANALOG SIGNALS ARE SCALED GE -1.0 AND LE +1.0.
C     INTEGER ANALOG SIGNALS ARE SCALED GE -32767 AND LE +32767(14 BIT).
C
C     PROGRAM VARIABLES
C        IERR            ERROR CODE FOR RESERVATION
C                        0=NOERROR, GT.0=RESERVATION ERROR
C        ISTAT           REAL TIME MODE
C                        0=IN REAL TIME, ISTAT.GT.0 NOT IN REAL TIME
C        OUT             THE DAC VARIABLE
C        BACK            THE ADC VARIABLE
C        PERCENT         MAXIMUM ALLOWABLE PERCENT ERROR
C        PERROR          ACTUAL COMPUTED PERCENT ERROR
C        LINE            NUMBER OF LINES OF PRINTOUT IN EXECUTION
C
C     DADIOS PATCHING REQUIREMENTS (AD/4 FIELD 3, 4TH GROUP ADC AND DAC)
C
C          TRUNKING              FORTRAN                 AD/4 LOGIC
C        W-03 TO V-06      FOR /*ADC1/49,ADC       TR10-TR17 AND TR30-TR37
C        W-13 TO V-07      FOR /*DAC1/49,DAC       TR50-TR57 AND TR70-TR77
C
C
      COMMON/INTCOM/OUT,LOOP,BACK,PERCENT,PERROR
      INTERRUPT(I=1,R=1,T=500)
      COMMON/*ADC1/49,ADC
      COMMON/*DAC1/49,DAC
C
C     INITIALIZATION
C
      PERCENT=.05
      OUT=0.0
      BACK=0.0
      LOOP=0
      LINE=0
      CALL RESERVE(IERR)
      WRITE(6,1000)IERR
      IF(IERR.NE.0)STOP
C
C     REAL TIME
C
      CALL SIMRUN(ISTAT)
      WRITE(6,2000)ISTAT
      IF(ISTAT.GT.0)STOP
      CALL REMARK(17H JOB IN REAL TIME)
   25 CONTINUE
      CALL QHOLD
      LINE=LINE+1
      IF(LINE.GT.200)STOP
      WRITE(6,4000)OUT,BACK,PERROR,LOOP
      CALL SIMGO
```

```
      GO TO 25
      WRITE(6,3000)
 1000 FORMAT(24H RESERVATION ERROR CODE=,O20)
 2000 FORMAT(18H REAL TIME STATUS=,O20)
 3000 FORMAT(1H0,*PROGRAM TERMINATED NORMALLY*)
 4000 FORMAT(5X,3F10.4,I10)
      STOP
      END


      SUBROUTINE SUB1
C
C     REAL TIME INTERRUPT SUBROUTINE
C
      COMMON/INTCOM/OUT,LOOP,BACK,PERCENT,PERROR
      COMMON/*ADC1/1,ADC
      COMMON/*DAC1/1,DAC
      DAC=OUT
      BACK=ADC
      PERROR=ABS((OUT-BACK)/OUT)
      IF(PERROR.GT.PERCENT)GO TO 10
      IF(OUT.GT.0.99)OUT=0.0
      OUT=OUT+.005
      LOOP=0
   10 LOOP=LOOP+1
      IF(LOOP.EQ.10)CALL SIMHOLD
      CALL SIMIDLE
      END


      TREE TRALGIC(0),SUB1(1)
      GLOBAL INTCOM
      END
```